DESIRE6G

D2.2: Functional Architecture Definition

HEU 6G SNS JU Project

Grant No. 101096466

Co-funded by
the European Union

Document properties

Document number	**D2.2**
Document title	DESIRE6G Functional Architecture Definition
Work Package	WP2
Editors	Gergely Pongrácz (ERI-HU)
Authors	Merve Saimler, Sultan Ertas (EBY), Anastassios Nanos (Nubis), Anestis Dalgkitsis, Cyril Hsu, Chrysa Papagianni (UVA), Luis Velasco, Marc Ruiz, Sima Barzegar, Jaume Comellas (UPC), Sándor Laki (ELTE), Kiran Chackravaram, Simon Pryor (ACC), Attila Mihaly (ERI-HU), Rafael Lopez da Silva, Marta Blanco Caamaño, Alejandro Muñiz Da Costa, Guillermo Sánchez Illán (TID), Vincent Lefebvre (TSS), Pablo Picazo (UC3M)
Internal reviewers	1. Chrysa Papagianni (UVA), 2. Sandor Laki (ELTE)
External reviewers	1. Carlos J. Bernardos (UC3M), 2. Francesco Paolucci (CNIT)
Dissemination level	Public
Status of the document	Final
Version	V5
File name	DESIRE6G_D2_2_Functional_Architecture_Definition
Contractual delivery date	30/6/2024
Delivery date	18/07/2024

Document history

Revision	Date	Issued by	Description
V1	29/01/2024	ERI-HU	Initial revision with ToC
V2	27/05/2024	ERI-HU	Version for internal review
V3	24/06/2024	ERI-HU	Version for external review
V4	01/07/2024	ERI-HU	Handling external review
V5	16/07/2024	ERI-HU	Submitted version

Abstract

The present deliverable specifies the architecture of the DESIRE6G system. This document serves as basis for the design and system integration of a next-generation network architecture aimed at meeting the stringent requirements of future 6G applications. The DESIRE6G architecture will apply three main innovations to move towards a more service-specific and dynamic behaviour. The system will be based on a flexible and dynamic service framework that enables applications to discover, instantiate, and utilize services on a per-request basis, promoting efficient resource utilization and simplified service management. It involves multi-timescale control loops considering various inputs, such as network topology and resource utilization, along with key performance indicators (KPIs) and policies (e.g., minimizing energy consumption). These loops enable infrastructure adjustments and multi-parameter optimizations with the support of AI. And finally, all of the DESIRE6G capable sites will run an infrastructure management layer. This layer acts as a combination of the Virtualized Infrastructure Manager (VIM) and a hardware abstraction layer (HAL), facilitating the use of hybrid hardware systems and cloud-native automation in handling physical resources without complicating the upper layers, especially the business logic. The DESIRE6G system is designed to integrate these innovations into a coherent architecture capable of achieving the project's KPIs and addressing diverse service and use case requirements. This document details the functional components, their interactions, and the processes within the DESIRE6G system, providing a roadmap for the deployment of an AI-powered, zero-touch, cloud-native 6G network that supports ultra-reliable low-latency communications (URLLC) and other critical 6G services

Keywords

Architecture definition, API definition, message flows, system view

Disclaimer

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101096466.

DESIRE6G is supported by the Smart Networks and Services Joint Undertaking. This report reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

Table of Contents

1. Introduction 16
2. High-level view of the architecture 18
2.1 Main components and layers 18
2.2 DESIRE6G Service, Network Slices and QoS 22
2.3 Service definition 24
2.4 DESIRE6G Multi-timescale Control Loops 26
2.4.1 Traffic management 27
2.4.2 Local optimizations 27
2.4.3 Service-wide optimizations 28
2.4.4 Global Optimizations 29
3. Service Management and Orchestration 31
3.1 Evolution of O-RAN 6G Common SMO for NBI/API Exposure 31
3.1.1 Open RAN '6G Common SMO' Evolution 31
3.1.2 O-RAN nGRG 'Common SMO' 33
3.1.3 3GPP Hierarchical Management and Orchestration 34
3.1.4 3GPP NWDAF Evolution for Federated Learning 35
3.2 Service deployment and lifecycle management in DESIRE6G 36
3.3 Intent-based service framework 39
3.3.1 Receive Business Request (SLA) 40
3.3.2 Translate from SLA to Intent 41
3.3.3 Creation of Service Description and Graph 42
3.3.4 Decompose the Service for Actual Deployment 43
3.4 Optimization Engine 44
4. Service optimization and assurance 47
4.1 MAS and MLFO 47
4.2 Service monitoring 51

4.2.1 Reference Scenario 52
4.2.2 Monitoring components interaction 55
4.2.3 In-Band Telemetry Case Studies 56
5. Cloud native data plane with IML 60
5.1 Network function model 60
5.2 Cloud-Native Data Plane Mechanisms 62
5.3 Network Service Deployment and Operation 65
5.4 Infrastructure Functions 66
6. Main functions and message flows 70
6.1 Bootstrapping 70
6.2 Service creation 71
6.3 Service deployment 73
6.4 Service attachment 78
6.5 Service Assurance - MAS 79
6.6 Service Assurance - SMO 82
6.7 Mobility 83
7. Mobile network functions 85
7.1 Radio Access Network 85
7.2 Core network 89
8. Additional architecture considerations 92
8.1 MAS Security 92
8.2 Transport network 93
8.3 TSN and DetNet extensions 95
9. Summary and conclusions 99
10. References 100
11. Annex I: An example local network service descriptor 105

List of Figures

Figure 1: DESIRE6G architecture and layers 18
Figure 2: An example multi-site DESIRE6G system 19
Figure 3: An example RAN site 22
Figure 4: Communication services provided by network slice instances [2] 22
Figure 5: Services, slices and QoS 24
Figure 6: Service description as network function graph 24
Figure 7: Service types (end-to-end vs. edge-to-edge) 25
Figure 8: DESIRE6G Multi-timescale Control Loops 27
Figure 9: Evolutions of Open RAN (control-plane) towards 6G 32
Figure 10: Hierarchal & Federated SMO/Agents for 6G O-RAN 'Common SMO' 33
Figure 11: O-RAN nGRG 'Common SMO' for Cross-Domain Network Digital Twin 34
Figure 12: 3GPP Service API exposure and distributed MDAS/MDAF 35
Figure 13: 3GPP Evolved NWDAF with Distributed Analytics 36
Figure 14: D6G SMO overview 37
Figure 15: Data Lake interfaces 37
Figure 16: Service Catalog Interface 38
Figure 17: Service Orchestrator interfaces 38
Figure 18: IBN architecture 40
Figure 19: Optimization Engine service partitioning automation. 44
Figure 20: Optimization engine module internal architecture. 45
Figure 21: Optimization engine AI/ML Workflow. 46
Figure 22: MAS and MLFO scheme 47
Figure 23: MAS overview 48

Figure 24: Illustrative Use case 49
Figure 25: Intelligent routing Use Case 50
Figure 26: Pervasive Monitoring architecture scenario 52
Figure 27: Interaction among montoring components 55
Figure 28: Pervasive telemetry cases of study 57
Figure 29: Two stage telemetry collector used for postcard telemetry implementation 59
Figure 30: Interaction of network function control and data planes via IML 61
Figure 31: Load optimizer component for seamless load balancing and hardware offloading 64
Figure 32: Multitenancy support with P4 Program aggregation (P4-AGG) 65
Figure 33: Local NSD describes a subgraph of the service graph 66
Figure 34: The main DESIRE6G header 67
Figure 35: First step for Starting a new DESIRE6G system 70
Figure 36: Adding a new site to the DESIRE6G system 71
Figure 37: The service creation process 72
Figure 38: The service deployment process 75
Figure 39: Intermediate logical view of the system during deployment 76
Figure 40: Last deployment step with the physical view of the system 76
Figure 41: User attaches to a selected service 78
Figure 42: Service reconfiguration performed by MAS 80
Figure 43: Partial or full redeployment of a service or services 82
Figure 44: Handover handled automatically 83
Figure 45: D6G Open RAN based SNPN context 85
Figure 46: D6G Open RAN user-plane CU-UP/DU NF breakdown [20] 86
Figure 47: 'Southbound' RAN Evolutions of Open RAN (control-plane) towards 6G 86
Figure 48: DESIRE6G RAN utilizing eBPF/XDP User/Data Plane Programmability 89

Figure 49: Decomposed UPF – business logic vs. infra modules 90
Figure 50: MAS agent security leveraging DLT and rewriting 93
Figure 51: High-level view with the SMO-SDNC interface 94
Figure 52: The ALTO-based interface for getting topology information 95
Figure 53: The DetNet architecture 97
Figure 54: Using DetNet for transport between DESIRE6G sites 97

List of Tables

Table 1: Tabular Format for XR Availability QoS 42

Table 2: UE2Service Mapper API calls 67

Table 3: NFRouter API calls 68

List of Acronyms

3GPP	3rd Generation Partnership Project
AAL	Acceleration Abstraction Layer
AF	Application Function
AI	Artificial Intelligence
AI/ML	Artificial Intelligence / Machine Learning
AIaaS	Artificial Intelligence as a Service
AP	Access Point
AR	Augmented Reality
ASIC	Application Specific Integrated Circuit
BWE	Bandwidth Enforcer
CFS	Control Flow Shadowing
CID	Connection ID
CODP	Communication and Dissemination Plan
CPU	Central Processing Unit
CU	Centralized Unit
CUDA	Compute Unified Device Architecture
DFP	Data Flow Programming
DLINT	Deterministic Lightweighted In-Band Network Telemetry
DLT	Distributed Ledger Technology
DNN	Deep Neural Network
DPU	Data Processing Unit
DT	Digital Twin
DU	Distributed Unit
E2E	End-to-End
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GPU	Graphics Processing Unit
GUI	Graphic Users Interface
HAP	High-Altitude Platform
HD	High Definition

HH	Heavy hitter
HLIR	High-Level Intermediate Representation
HQoS	Hierarchical Quality of Service
HW	Hardware
IML	Infrastructure Management Layer
INT	In-band Network Telemetry
IoT	Internet of Things
ITU-T	International Telecommunication Union Telecommunication standardization sector
K8S	Kubernetes
KPI	Key Performance Indicator
KVI	Key Value Indicators
L4S	Low Latency, Low Loss, Scalable Throughput
LEO	LEO
LO-DP	Load Optimizer data plane
MAS	Multi Agent System
MIMO	Multiple-Input and Multiple-Output
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
MR	Mixed Reality
NF	Network Function control plane
NF-DP	Network Function data plane
NFR	Network Function Routing
NGMN	Next Generation Mobile Networks
NIC	Network Interface Card
NN	Neural Network
NS	Network Service
OAM	Operations, Administration and Maintenance
O-RAN	Open RAN
OVS	Open vSwitch
P4	Programming Protocol-Independent Packet Processors
PDP	Programmable Data Plane
PLINT	Probabilistic Lightweighted In-Band Network Telemetry

PM	Performance Monitoring
PoC	Proof of Concept
PPV	Per Packet Value
QoS	Quality of Service
RAN	Radio Access Network
RIC	Real Time Intelligent Controller
RNN	Recurrent Neural Network
ROS	Robot Operating System
RoT	Root of Trust
RTK	Real-Time Kinematic positioning
RTT	Round-Trip Time
SDG	Sustainable Development Goals
SDN	Software-Defined Networking
SDN-CP	Software-Defined Networking Control Plane
SECaaS	Security as a Service
SLA	Service Level Agreement
SLAM	Simultaneous Localization and Mapping
SMO	Service and Management Orchestrator
SNS IA	Smart Networks and Services Infrastructure Association
SNS JU	Smart Networks and Services Joint Undertaking
SO	Service Orchestrator
SRv6	Segment Routing IPv6
SW	Software
TRL	Technology Readiness Levels
UAV	Unmanned Aerial Vehicle
uCID	Micro Connection ID
UE	User Equipment
uFC	Micro Flow Cache
uRLLC	Ultra-High Reliability & Low Latency Communications
VES	VNF Event streaming
VNF	Virtual Network Function
VR	Virtual Reality

WAN	Wide Area Network
XR	Extended reality

Executive Summary

The overarching goal of DESIRE6G is to design a zero-touch control, management, and orchestration platform with native AI integration, supporting extreme URLLC application requirements over a performant, measurable, and programmable data plane. This will be achieved by assembling/utilizing different enabling technologies (from the data plane up to the orchestration plane) that are expected to constitute the foundation for the 6G architecture; namely, data plane programmability and deep slicing, AI, blockchain and cloud-native, serverless computing. The key innovations of DESIRE6G are summarized in the following.

- DESIRE6G introduces the use of a fully programmable, highly disaggregated RAN, transport and core forwarding plane that facilitates flexible, customized packet processing operations and protocol support, over heterogeneous devices such as ASICs, GPUs etc.
- It allows for a unified view of that programmable data plane, seamlessly managing scaling operations, hardware acceleration, and function deployment, while abstracting the complexity of handling the physical resources.
- By supporting multi-tenancy in the programmable data plane, DESIRE6G stretches the deep slicing concept further to the user plane, allowing slice-specific SW stacks and HW acceleration. On a per-service basis, customized control and protocol support can be the market differentiator, enabling novel application-network interactions and enhanced performance.
- The DESIRE6G architecture follows a cloud-native approach for mobile network deployment, leveraging cloud-native technologies, encompassing serverless computing principles and advanced infrastructure management, introducing cloud-native data plane mechanisms.
- DESIRE6G is re-architecting mobile network management, realizing autonomic networking through a hybrid, centralized management and orchestration architecture, with distributed intelligence closer to the physical infrastructure that is relying on the use of Multi-Agent Systems (MAS). The introduction of a common service management and orchestration framework for the RAN, Core and Transport, is in line with the emerging consensus in standards development organizations and fora that an efficient and effective 6G architecture needs to be considered cross-domain.
- DESIRE6G's AI-native architecture integrates AI across all layers of the network, from centralized orchestration and management to MAS-based distributed intelligence and edge intelligence. This comprehensive integration of AI enables adaptive, fast decision-making, optimized

resource management, and improved quality of service, making it a cutting-edge solution for future 6G networks.

- The DESIRE6G Monitoring system enhances network visibility, stretching from the core network to the user equipment, utilizing network telemetry solutions for accurate end-to-end information collection. The use of AI and data plane programmability is pivotal for providing dynamic and flexible network telemetry. Furthermore, the monitoring system is integral to DESIRE6G AI pipelines as it provides the necessary data quality, real-time insights and comprehensive coverage needed for effective AI model training, validation, and deployment.
- DESIRE6G employs distributed ledger technology as a zero-trust mechanism at different levels of the hybrid architecture (i) to enable dynamic federation between the different administrative domains and (ii) to secure the proposed MAS.

This document presents the functional architecture of the DESIRE6G system, which is designed to address the challenges of future 6G networks. It outlines its key components, their functionalities, and the interactions within the DESIRE6G architecture, highlighting the above-mentioned innovations.

Key Contributions

The key contributions of the deliverable:

- The functional definition of the DESIRE6G architecture spanning from the physical infrastructure to network control and end-to-end lifecycle management mechanisms required to support the set of 6G services and requirements, as identified in Deliverable 2.1.
- Identification of the Multi-Timescale Control Loops of the system.
- The description of the main architectural modules, their logical connections, and the required information exchanges.
- The description of the main functions and message flows within the DESIRE6G system.
- Proposals for RAN acceleration and Core Network disaggregation.
- The description of integration with external components (e.g., SDNC, 3GPP control plane services, physical network functions).

1. Introduction

In the 5G era, the mobile communication ecosystem aimed to tackle multiple challenges using the same network infrastructure, primarily optimized for the most prevalent use case: mobile broadband (MBB), which primarily facilitates Internet access for mobile phones and other devices. However, the introduction of URLLC (Ultra-Reliable Low Latency Communications) was essential for supporting latency-sensitive services (i.e., Industry 4.0 applications, etc.). URLLC came with quite strict guarantees and thus quite rigid resource consumption which in some cases led to low resource utilization.

Beyond real-time mission-critical applications, mobile broadband traffic itself is quite diverse. Some data flows, like video conferencing and online gaming, are latency-sensitive, while others prioritize bandwidth and overall flow completion time. Currently, network traffic is predominantly encrypted end-to-end, obscuring both payload and often the header, which complicates traffic management based on traditional packet-handling behaviours. Thus, the question arises: how to best support these diverse traffic classes in the current network.

Historically, network traffic management was dominated by two systems: circuit-switched, used mainly by the telecom industry, and packet-switched, favoured by the IT industry. These methodologies have long been at odds, evident in past conflicts like ATM vs. IP and MPLS vs. Ethernet. Currently, the challenge resides in industrial settings, where there is a demand for nearly wire-like reliability without the physical constraints of cables. In these scenarios, DetNet and TSN represent the circuit-switched approach, while class-based QoS and latency-aware AQM methods represent the packet-switched response.

The question remains whether these approaches are opposing or complementary. The network must efficiently and economically accommodate diverse traffic types, even when traditional KPIs like latency need to be exceptionally stable yet minimal. Achieving ultra-high bandwidth with sub-millisecond latency and five-nines reliability is currently not feasible. What the network should support is optimal use of the available resources for applications at the UE. Effectively, with the same set of resources, it could lead to a quite different quality of experience. This is where we need network programmability. In many cases simple schemes have proven effective, like using shallow buffers and limited RAN retransmissions for latency-sensitive traffic. Further steps can be taken in that direction, such as using over-protected services with strong FEC and potential header compression for ultra-low latency flows to minimize jitter from RAN retransmissions. While achieving five-nines reliability may still be out of reach, these measures can ensure constant and ultra-low latency. In the case of congestion, value-based packet dropping is often sufficient. However, as a last resort, resource reservation can be implemented

relatively easily if the appropriate APIs are in place. This bridges the gap between the simple and efficient "packet-switched" and the more deterministic and predictable "circuit-switched" worlds.

To ensure the successful implementation of the above, we need an appropriate 6G system architecture. DESIRE6G attempts to address performance requirements for real time mission-critical applications by designing and prototyping an AI-powered, intent-based, cloud-native 6G system architecture supported by a performant, measurable, predictable, and customizable data plane. Towards this direction, DESIRE6G specifies:

- A **flexible and dynamic service provisioning framework**, where the applications can easily find the proper service needed for them and that service can be easily instantiated even dynamically on a per-request basis. The framework should hide the complexity of handling such services.
- **Multi-timescale control loops** that can consider various inputs (e.g., network topology, resource utilization), KPIs and policies (e.g., minimize energy consumption) to request changes in the infrastructure make multi-parameter optimizations, with AI support.
- An **infrastructure management layer** that can separate concerns of the business logic and the infrastructure acting as a combination of the Virtualized Infrastructure Manager (VIM) and a hardware abstraction (HAL) layer. This approach allows for the use of hybrid hardware systems or cloud-native-like automatisms in physical resource handling, without overcomplicating the upper layers, particularly the business logic.

The DESIRE6G architecture is based on the O-RAN principles (see section 3.1), but is defined to extended to meet the aforementioned goals. The document is organized as follows. The functional architecture of the DESIRE6G system is outlined in Section 2. Sections 3 to 5 describe the functionalities of the main modules, their logical connections, and the required information exchanges. Section 6 details the main functions and message flows within the DESIRE6G system, including processes such as bootstrapping, service creation, deployment, and reconfiguration. Section 7 focuses on the implementation of basic mobile network functions within the DESIRE6G system, including details on the Radio Access Network (RAN) and core network components. Finally, Section 8 discusses additional architecture considerations, including security aspects, transport network requirements, and extensions like TSN and DetNet within the DESIRE6G system.

2. High-level view of the architecture

This section contains some basic assumptions, terminology clarifications and the main building blocks and their connections.

2.1 Main components and layers

The main components of the DESIRE6G system can be seen on Figure 1 with four basic layers. Note that the D6G architecture re-uses/adopts many components and concepts from previously existing architectures, some defined by SDOs (e.g., ETSI NFV, O-RAN).

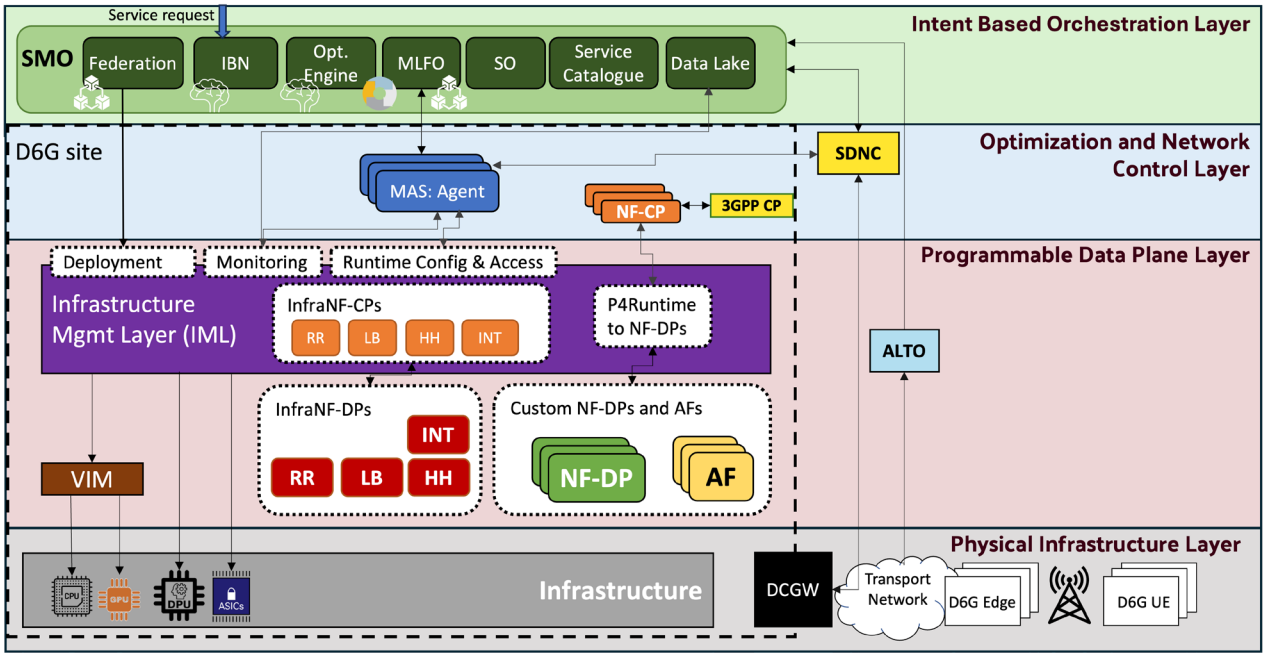

FIGURE 1: DESIRE6G ARCHITECTURE AND LAYERS

Figure 1 shows one DESIRE6G site in detail. DESIRE6G is a multi-site system, where inter-site traffic will likely traverse external components (e.g., normal switches and routers). To successfully integrate these "non-D6G entities" we need to cooperate with their controllers (e.g., SDNC). The User Equipment (UE) might also be able to run a stripped-down version of the DESIRE6G stack using its own hardware and software. This could be especially important for implementing end-to-end services and controlling their KPIs. See an example sketch with multiple DESIRE6G sites (including a D6G UE) on Figure 2.

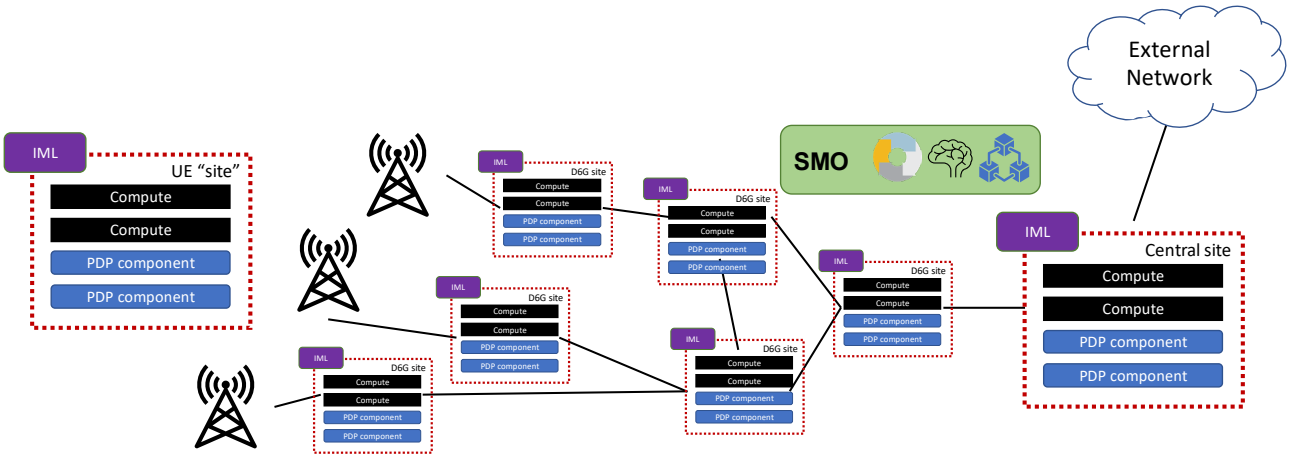

FIGURE 2: AN EXAMPLE MULTI-SITE DESIRE6G SYSTEM

The details of the different layers, their main modules and functionality are summarized below following a "top-down" view, while a more detailed description is provided in the following sections.

- **Intent-based Orchestration layer.** It comprises the Service Management and Orchestration (SMO) framework, which is responsible for orchestration, lifecycle management and automation of E2E network services (NSs), in line with current specifications (i.e., O-RAN [1], ETSI MANO). The D6G SMO focuses on innovations pertaining to ML-powered intent-based service management (IBN), orchestration of AI/ML workflows (MLFO), network service Federation employing Distributed Ledger Technology (DLT) and data management and exposure services pertaining to the SMO. Its northbound API enables the full operation of system and the NSs. In detail, the SMO is composed of the following components:
    - **Intent-based Service Management (IBN):** it is the module that translates high-level intents derived from SLAs and customer values to service templates and uploads them into the Service Catalog.
    - **Service Catalog:** it stores the service descriptions in the form of service graph(s) – one per direction – and related parameters, e.g., QoS requirements.
    - **Service Orchestrator:** it is the key entity at service deployment. Orchestration might take place statically before UEs start using the service, but there is also an option to make (partial) deployment when a user is attaching to the given service – we will call this the "dynamic" case. The orchestrator has knowledge of the operator's D6G sites and inter-site connections.
    - **Optimization Engine:** it is the entity responsible for generic optimizations on medium to long timescales (larger than 1 sec).

- **Machine Learning Function Orchestrator (MLFO):** it orchestrates, manages and monitors all the elements and resources that build an AI pipeline related a particular network service.
- **Data Lake** and **Federation:** they are support modules for storing data and for managing multi-site deployments. We assume that federation for the SMO is an internal process of the SMO logic and is described in detail in Deliverable 3.1 [26].

- **Optimization and network control layer.** This is the layer where control plane (CP) logic and optimization logic reside.
  - **Control plane logic:** it is needed for almost all data plane components. The CP logic terminates external protocols and interfaces (such as 3GPP interfaces, e.g., N4) and configures the NF data plane (DP) tables and other structures.
  - Service optimization is achieved mainly by the **multi-agent-system (MAS)** which implements distributed network intelligence closer to the physical infrastructure. MAS is responsible for receiving service-specific monitoring information and fine-tuning the network and compute resources to meet service-level KPIs. It configures and uses the pervasive telemetry system to receive service performance indicators, e.g., end-to-end latency for latency-sensitive or latency-critical services.
  - Although most of its components belong to the PDP layer, **pervasive monitoring** logically also belongs here as a main input generator for MAS via the telemetry agents.
  - External control plane entities, such as **SDN Controller** (SDNC) and **3GPP Control Plane** entities also belong to this layer. Note that in all architecture figures the yellow colour code means that the given entity or module is seen as an external entity from the DESIRE6G perspective, which is integrated to the architecture, but taken "as is".
- **E2E Programmable Data Plane (PDP) layer.** This layer is the execution layer for the services. It implements flexible, customized packet processing operations and protocol support.
  - The E2E programmable data plane is employing **Network Functions** (NFs) to carry out the logic of the selected service. Most NFs split into a Control Plane (NF-CP) and Data Plane logic (NF-DP). Note that at this level the main role of NF-CP is to configure and update the objects (e.g., tables, registers, etc.) in the corresponding NF-DP(s). Different NF types exist in the D6G system:
    - **Business logic NFs** (green and yellow) execute packet processing steps that are dictated by the given service including application logic.

        - **Infrastructure NFs** (red) execute steps that are done automatically and independently from a given service. Typically, these are common functions used for every service, like NF-NF routing (for executing the service graph), load balancing, telemetry collectors or transport adapters. See details in section 5.
    - **Infrastructure Management Layer (IML).** It is responsible for transparent scaling, hardware acceleration, and function deployment, i.e., it is the bridge between the logical and the physical network function. The IML hides the implementation and deployment details of the NF-DP from the NF-CP by providing a simple unified view of the data plane. For example, even if NF-DP is vertically or horizontally disaggregated into multiple data plane programs running on different HW/SW targets, IML ensures that NF-CP only sees a single NF-DP instance and handles the complexity of managing the disaggregated packet processing components.
    - **Virtual Infrastructure Management (VIM).** It is responsible for managing the compute resources of the D6G sites. In the project the use of Kubernetes is planned for this functionality, but as a logical function one could envision using other VIMs too (e.g., OpenStack, VMWare)
    - **Pervasive monitoring** is achieved by functionalities that span by multiple components, i.e., monitoring probes injected in the PDP and components deployed as infrastructure NFs, monitoring capabilities of the VIM, UE monitoring etc.
- **Physical Infrastructure Layer.** The physical infrastructure contains the compute and networking components of the D6G system and some "black box", non-programmable hardware entities or fixed function programs that are used by different functions, like transport network nodes or RAN hardware blocks. Different sites might have quite different hardware setups, e.g., RAN sites will probably contain DSP hardware for signal processing and they will also run some low-latency control modules, such as nRT-RIC and CU-CP as shown in Figure 3, while core sites will typically contain more compute resources and possibly generic accelerators like GPUs and SmartNICs, as shown in Figure 1.

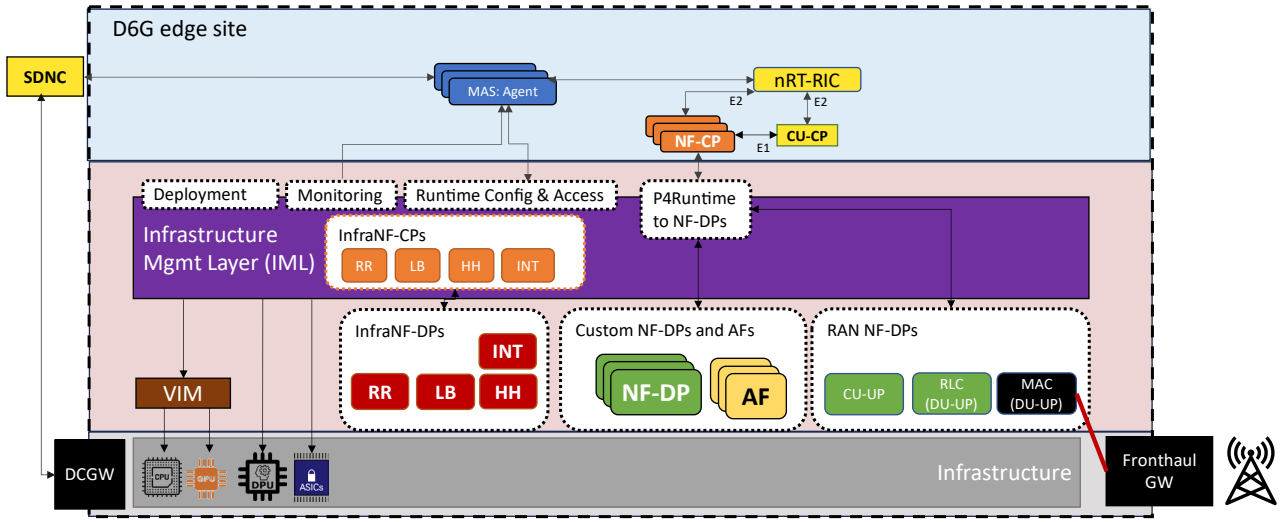

FIGURE 3: AN EXAMPLE RAN SITE

2.2 DESIRE6G Service, Network Slices and QoS

According to ETSI TS 128 530, clause 4.1.3 [11] communication services are supported by network slices, while a typical end-to-end network slice spans from the RAN to the core segment, as shown in Figure 4 . On the core side, it is usually composed by the "network functions" and their interconnections. Such network functions can be implemented through VNFs or PNFs that are combined and interact among each other through several Virtual Links (VLs) to comprise Network Services.

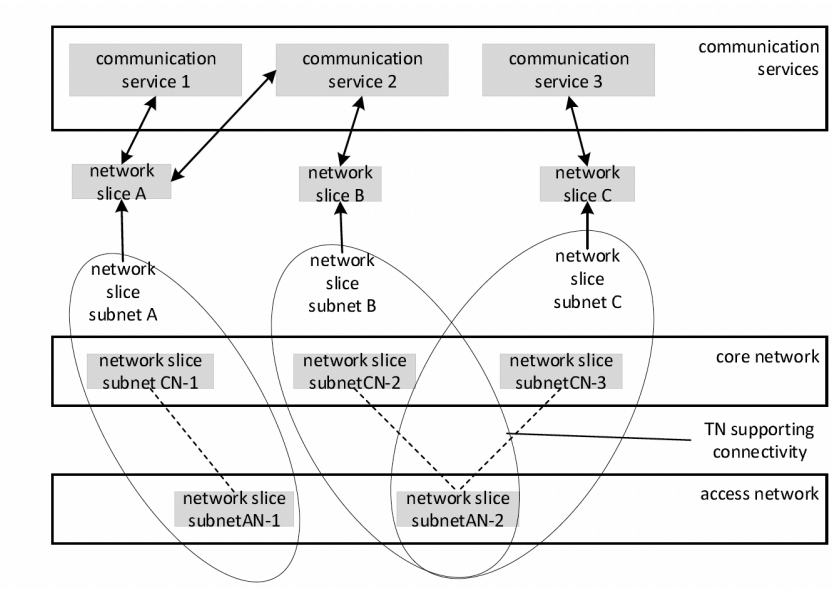

FIGURE 4: COMMUNICATION SERVICES PROVIDED BY NETWORK SLICE INSTANCES [3]

According to ETSI GS ZSM 003 [3], a vertical may have multiple communication services and one or more communication services may be supported by one network slice instance as shown in ETSI TS 128 530.

In DESIRE6G we assume that each D6G (communication) service is supported by a network slice instance, and inside this slice / service various QoS classes can be defined freely. Consequently, two levels of QoS are supported in DESIRE6G:

1. The highest level in this hierarchy represents services/slices, with dynamic resource sharing ratios.
2. Each network slice can define multiple QoS classes with additional resource sharing rules. Users mapped in a QoS class can be handled differently (e.g., by assigning different weights to them).

Thus, in DESIRE6G traffic management needs to handle Hierarchical QoS (HQoS) policies. On all layers of the hierarchical QoS scheme, the usual resource sharing method is weighted fair queuing, e.g., set the bandwidth ratio between two slices to X:Y. Other methods can also exist, e.g., strict priority for a given slice – typically used for "mission critical" services or more complex schemes.

In the DESIRE6G system, slices are isolated from each other in terms of performance. For a network slice the system should define:

- The service graph and its corresponding KPIs (e.g., end-to-end latency, throughput etc).
- Depending on service deployment, the respective sites and resources that are available for the given service (optional).
- The QoS policy for individual users inside a given slice (optional).

The QoS policies are defined together with the service (see section 2.3). The service-specific QoS definitions will be collected both by MAS and IML. MAS will be in charge of taking system-wide decisions, (e.g., optimal service-specific routing), while IML will configure the low-level Traffic Management / queuing policies and optimizing load. Figure 5 shows the relation of QoS and services.

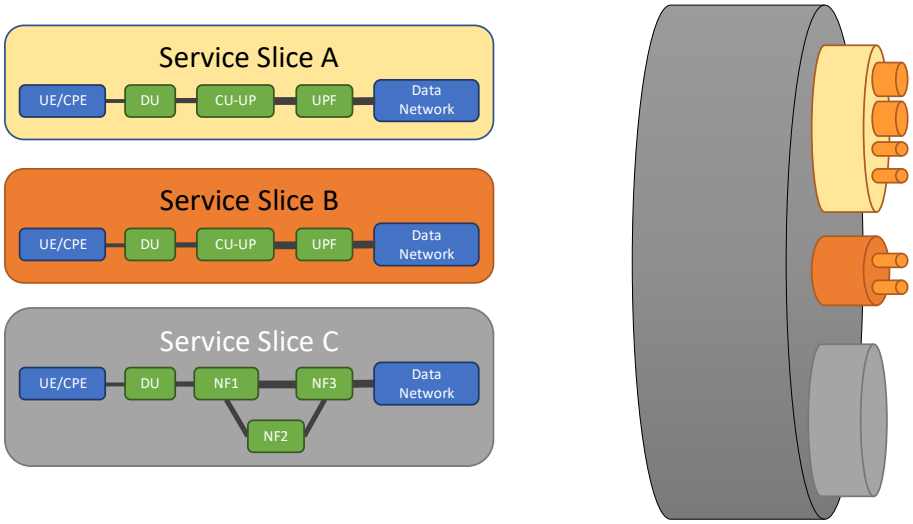

FIGURE 5: SERVICES, SLICES AND QOS

2.3 Service definition

Service in the D6G context equals to the end-to-end behaviour that system provides to the end users. The service is described by a service graph (or NF-graph), as shown in the figure below.

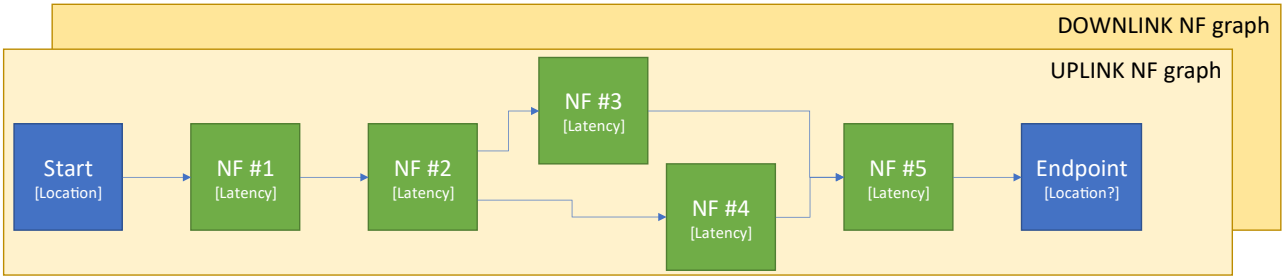

FIGURE 6: SERVICE DESCRIPTION AS NETWORK FUNCTION GRAPH

The service graph is described for both uplink and downlink direction, as there could be different behaviour based on the direction. In some cases, it is preferable to collocate UL and DL handling to the same physical instance of a given NF. This can be achieved by setting the same user identifier in both directions (denoted as load balancing key and detailed in Section 5.2), ensuring the selection of the same "NF worker" or "NF-DP".

In D6G there are two types of services as depicted on Figure 7:

- End-to-end: the service graph includes application workloads (e.g., UE app – edge app), meaning that the entire end-to-end path is controlled by the D6G system. This is preferable for latency critical applications, e.g., the AR/VR DESIRE6G use case.

- Edge-to-edge or UE to UPF. This means that the app and the server are not part of the service graph. This is the "legacy" service, e.g., Internet access, where only the access part is controlled by the operator.

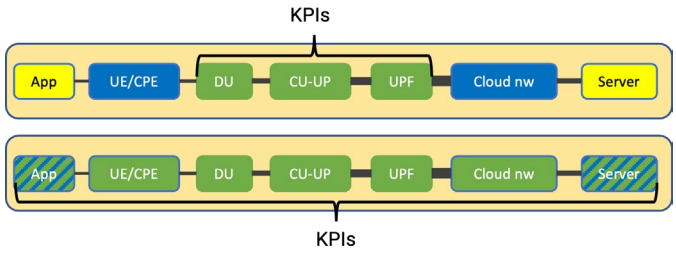

FIGURE 7: SERVICE TYPES (END-TO-END VS. EDGE-TO-EDGE)

Besides defining the building blocks (NFs, AFs) and their connections, the service definition may also contain requirements on the services or the NFs, such as:

- End-to-end (or edge-to-edge) QoS requirements:
  - Latency: a number (in msec – optional) and the behavior ("mandatory" or "preferred"). There can be multiple latency sensitive services.
    - In some cases, there are strict latency requirements (e.g., robot control), which means the service definition will contain an exact latency value and the "mandatory" flag.
    - In other cases, the application knows the exact value, but if that is not met the service can still work with some Quality of Experience (QoE) degradation (e.g., live streaming). These services are best described by the latency value where there is no QoE disruption and the "preferred" flag.
    - In some other cases, the application will only know that it would prioritize low latency over other KPIs but won't be able to specify an exact number. In these cases, no number will be present but a "low latency preferred" flag will be set.
  - Per flow data rate (for each UE connected to the given service): similarly to latency, a number (in bps) and the behavior ("mandatory" / "preferred") can be set.
    - Typically, a "preferred" flag is used, that is ideal for most of the services that download or exchange some information, e.g., web, social media or video on demand.
    - In some cases, the service requires a certain minimum bandwidth, e.g., video streaming. Sometimes this can be negotiated, e.g., if the available rate is smaller,

the streaming application can switch to a lower quality format. For such services a certain bitrate and "mandatory" flag can be specified.
        - The previous application might want to specify a "preferred" and a "mandatory" value too, showing its capability to adapt, but also its lower limits for adaptation.
        - There are cases when there is no preference at all, e.g., "lower-than-best-effort" type of services used for background updates.
    - Service reliability or tolerated packet loss: although most of the services won't specify exact numbers, it should be possible to ask for some guarantees, e.g.:
        - "Lossless": meaning the traditional 5 9s.
        - "Out of order packets": for legacy services it might be needed (set to zero), in normal cases it is not set, but maybe fine-tuning of some services will require some limit here.
        - Exact reliability number (e.g., 99%): in some cases, exact numbers can be set, e.g., a streaming video service with well-known codecs can set this according to the codec's tolerance.
        - "Best available": the system could calculate a value based on the other KPIs and the network condition and signal it during the attachment step.
        - If nothing is set (default): other KPIs will take precedence, but the system will try to minimize loss nevertheless (so the service will behave the same as in the "best available" case).
- UE to NF latency: even some intermediate NFs are needed to be close to the UE that they serve (e.g., RAN MAC, RLC). The system must have a way to describe this in the configuration of the service.
- Other policies, e.g., regional availability, time of day availability could also be considered.

2.4 DESIRE6G Multi-timescale Control Loops

In DESIRE6G different control loops are utilized to manage and optimize network performance at various levels and timescales. In particular, four distinct DESIRE6G layers are controlling automatically resources, functions and processes under their jurisdiction, instructing lower layers to gather information and to modify settings or deployments. Timescale of optimization is different between the layers. The timescale of these four layers and their associations are depicted in Figure 8.

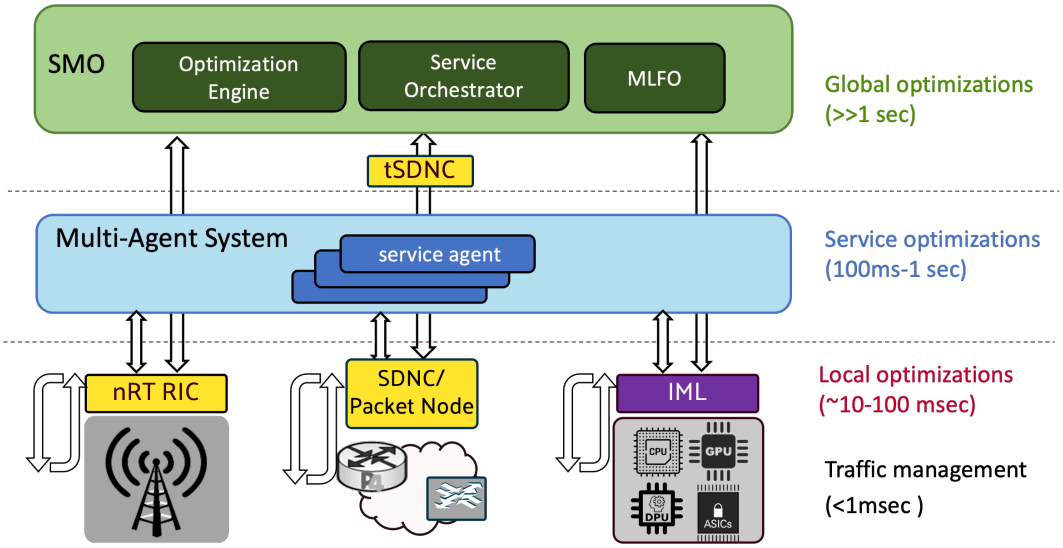

FIGURE 8: DESIRE6G MULTI-TIMESCALE CONTROL LOOPS

2.4.1 Traffic management

This is the fastest control loop operating on packet level at real-time. The main goal of the TM level is to achieve the optimal utilization of the bottleneck resources via smart queuing and dropping policies.

Control is done via proper configuration of the TM modules. The configuration is done according to the QoS parameters of the services active on the given TM entity. In DESIRE6G we will use novel TM concepts to make the multi-service QoS policies easy to implement, reported in the deliverable D4.1 [42]. The main idea is to calculate the "value" of a given packet and use this when there is a bottleneck to drop the "least important" packets from a given queue. This concept allows the easy definition of different TM schemes and allows quite complex multi-service, multi-user TM settings without adding too much complexity to the transport nodes.

2.4.2 Local optimizations

The second lowest layer is the local infrastructure (RAN or DC), where control is domain and resource specific. Each domain controller does local optimizations and tries to solve local problems, e.g., resource outages or overload situations.

Following the O-RAN architecture [1] the radio resources are controlled by the respective RAN **Central Unit - Control Plane** (**CU-CP**) as part of the disaggregated O-CU, supported by the **Near-Realtime RIC** (**nRT-RIC**) - which is optional. The nRT-RIC operates control loops in the range of up to 10 msec, executing local optimizations based on policies like power consumption

optimization and has accurate information on radio resource utilization. Note that these controllers can be seen as external components for the DESIRE6G architecture, they are taken directly from the O-RAN architecture

- Compute and network resources are controlled by the **IML**, as reported in the Deliverable 4.1 [42]. Specifically, the IML is handling D6G intra-site computing resources via the deployed Virtualized Infrastructure Manager – VIM, and programmable hardware units via P4Runtime interfaces. IML controls site internal run-time optimizations (e.g., scaling and heavy-hitter offloading) and have accurate information on resource situation inside a given DESIRE6G site.
- SDNC / Packet Node controls transport links and can do quick and local optimizations, e.g., applying fast reroute if a given link goes down. Note that this component is external to the DESIRE6G architecture and its use is optional.

The infrastructure controllers can do their automatic reactions in the order of tens or hundreds of milliseconds, so on the near real-time timescale.

2.4.3 Service-wide optimizations

The third layer is based on the MAS (see details in section 4.1), which supports end-to-end service optimization across the different technology domains. MAS agents collect information from the lower layers, e.g., resource utilization from nRT RIC or IML and they can also deploy telemetry functionality for given services to be able to check E2E and service specific KPIs that are not visible for a given low-level controller (see more details in the monitoring / telemetry section later). MAS can also instruct the lower layer controllers to optimize local behaviour, in order to meet E2E service specific KPIs, using their respective northbound interfaces.

While the envisioned MAS supports intelligent decision-making on a per service basis, it may occasionally encounter issues that require more comprehensive, network-wide resource management. In cases where the issue is significant (such as continuous KPI violations), MAS can escalate the problem to the highest layer (SMO) to initiate a partial or full service redeployment.

The MAS-based optimization loop is executed in the below one second timescale where the exact value depends on the nature and size of the problem. Thus, the O-RAN mandated near-real-time control loop (i.e. in the timeframe >10 ms and <1 s) [1], in DESIRE6G is split between the service and local -level optimizations, enhancing the network's responsiveness, efficiency, scalability, robustness, granularity, adaptability, and KPI management. Local-level optimization (10 ms - 100 ms) allow rapid, short-term

adjustments to maintain stability and performance. This ensures immediate reaction to dynamic network conditions, improving user experience and reducing latency. Service-level optimizations (100 ms - 1 sec) at the MAS level enable more complex adjustments that require a slightly longer timeframe, such as resource allocation, load balancing etc.

2.4.4 Global Optimizations

The topmost layer in optimization is taking place at the SMO framework, which can react to problems if it gets a trigger from the lower layers. The E2E SMO itself will not run optimizations automatically. Input triggers will mostly come from the MAS layer, but additional triggers are also possible, e.g., if a site goes down, the SMO needs to be aware of it. The SMO reaction speed is in the order of seconds, so it is the slowest optimization loop. The E2E D6G SMO is thus aligned with the O-RAN SMO framework where the non-RT RIC supports control loops in the order of seconds (>1sec) .

The transport network between DESIRE6G sites is – in most cases – controlled by the SDN controller(s) (tSDNC), providing a centralized platform to manage and control the entire transport network. Thus, problems that cannot be handled by the service level MAS are escalated to the SMO and the respective SNDS(s) that maintain accurate topology information and network state. Again, these controllers are not DESIRE6G specific, but are integrated with the rest of the architecture.

The introduction of the MAS layer, for the purpose of supporting service-layer optimizations across different domains while coordinating when needed with the SMO for more comprehensive, network-wide resource management, reduces the respective load on the SMO, enhancing the intelligence and scalability of the orchestration framework.

The next 3 sections describe the components of the system one by one focusing on the main functionality of the given modules and their external interfaces towards other modules. These sections are organized similarly as the layers on Figure 1.

- The orchestration layer is described in section 3
- The optimization and control layer is described in section 4
- While section 5 contains the programmable data plane layer

More details on the modules can be found in Deliverable 3.1 (SMO and MAS) and Deliverable 4.1 (PDP and telemetry).

The rest of the sections describe the main system functions (section 6), the implementation of basic mobile network functions (section 7), and few additional components and external relations (section 8).

3. Service Management and Orchestration

3.1 Evolution of O-RAN 6G Common SMO for NBI/API Exposure

DESIRE6G architectural design coexists with parallel 6G evolutions from a number other areas, such as recent 3GPP work (5G-Advanced studies and work items in Release-18 and Release-19), ITU-R (IMT-2030 [5]), O-RAN nGRG [6], numerous parallel R&I projects (for example, as recently presented to 3GPP on 6G use-cases [7] and the European Smart Networks and Services Joint Undertaking [8]) as well as numerous other relevant complementary activities within ETSI (such as TeraFlowSDN [9], ZSM [10], OpenSlice [11], tmforum [12] and many more.

While DESIRE6G provides specific research results and innovations towards 6G, without attempting to provide a final 6G/IMT-2030 architecture, due consideration is given to ensure alignment and consistency with the evolving 6G consensus, ensuring the project results will positively impact the future 6G standards.

3.1.1 Open RAN '6G Common SMO' Evolution

From a perspective of architectural evolution of a 5G Open RAN, based on O-RAN Alliance [13] extensions to 3GPP towards 6G, there are several pertinent drivers and evolutions, shown below in Figure 9, especially in the RAN control-plane.

To better explain the evolutions portrayed in Figure 9, these 6G evolutions to 5G Open RAN can be broadly group into:

- "Southbound RAN evolutions", with changes within the RAN NFs themselves (CU/DU/RU). These are elaborated later in section 7.1.
- "Northbound RAN evolutions" towards NBI/API integrations of RAN with other domains (Core, transport) and 'Common SMO', described in this section.

Within the control-plane, a lot of the evolution beyond addition of additional functionality (like ISAC) considers telemetry export and training/deployment frameworks to support advanced AI/ML intelligence throughout the RAN, from the lower levels (such as embedded AI/ML in PHY) to the higher layers (such as automated zero-touch operation).

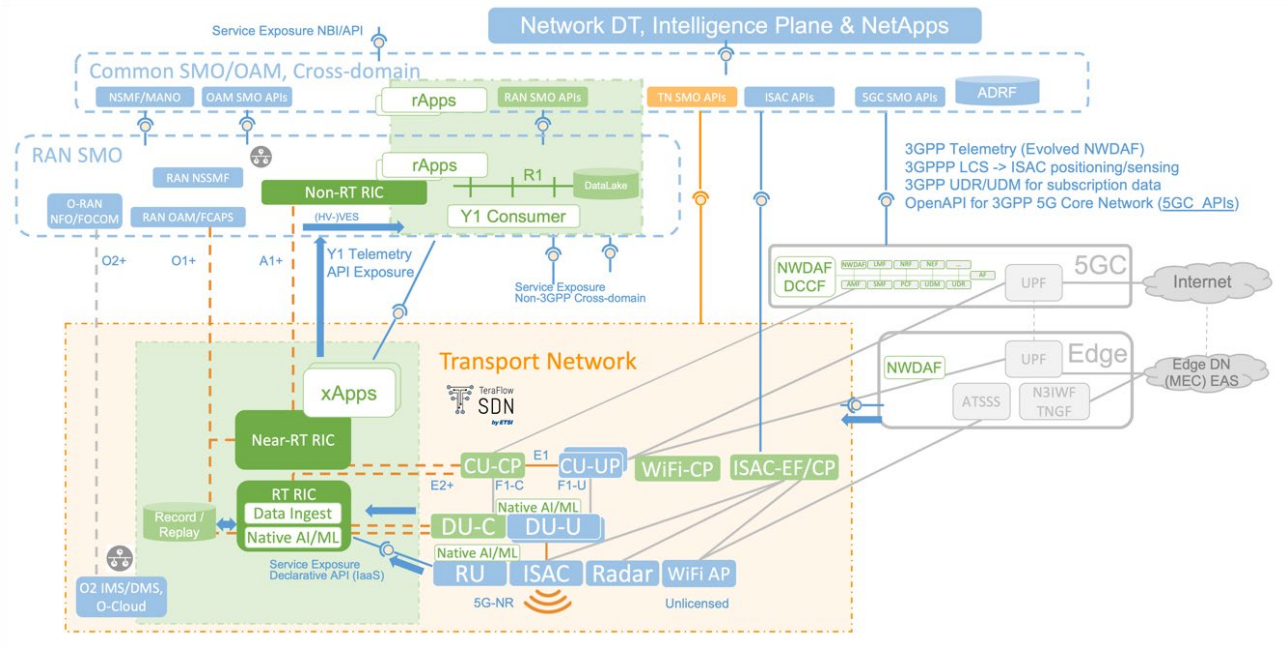

FIGURE 9: EVOLUTIONS OF OPEN RAN (CONTROL-PLANE) TOWARDS 6G

Whereas the 5G Open RAN SMO only considers the RAN itself, there is an emerging consensus that and efficient and effective 6G architecture needs to be considered cross-domain (RAN, Core and Transport), so the higher layers (termed SMO by O-RAN) require integration into a '6G Common SMO':

- O-RAN nGRG 'Common SMO'
- 3GPP Hierarchical Management and Orchestration
- 3GPP NWDAF Evolution for Federated Learning, see 3GPP TS 23.288 v18.6.0 [19].

For integration with existing higher-layer frameworks, like ETSI OpenSlice [11] or ETSI ZSM [10], it is assumed that some or all of the cross-domain integration (RAN, Core, Transport) will occur within those frameworks. However, a more integrated standalone O-RAN 'Common SMO' can be considered, where there is a merging of the integration of the RAN & Common SMOs, with the current non-RT RIC and rApps being extended to the cross-domain layers, elaborated further in section 3.1.2. Providing programmable extensibility has been a key advantage and driver for adoption of the O-RAN Near-RT RIC with xApps, and increasingly to provide intelligence into B5G networks, such as for energy efficiency goals. This cross-domain integration and extensibility of the 'common SMO', with API exposure and non-RT RIC rApps will be a fundamental evolution of the O-RAN 5G SMO, to provide 6G capabilities such as Native AI/ML, zero-touch automated operation and Network Digital Twins (NDT).

One aspect of integrating AI/ML intelligence throughout a converged SMO is the scalability restrictions of a purely centralised SMO, due to the bottleneck of sending all telemetry to a central SMO layer (e.g. for AI/ML training). It has been demonstrated that a scalable architecture needs to support a distributed federated architecture, as also seen in the evolution of the 3GPP NWDAF Evolution for Federated Learning in section 3.1.4 and [14].

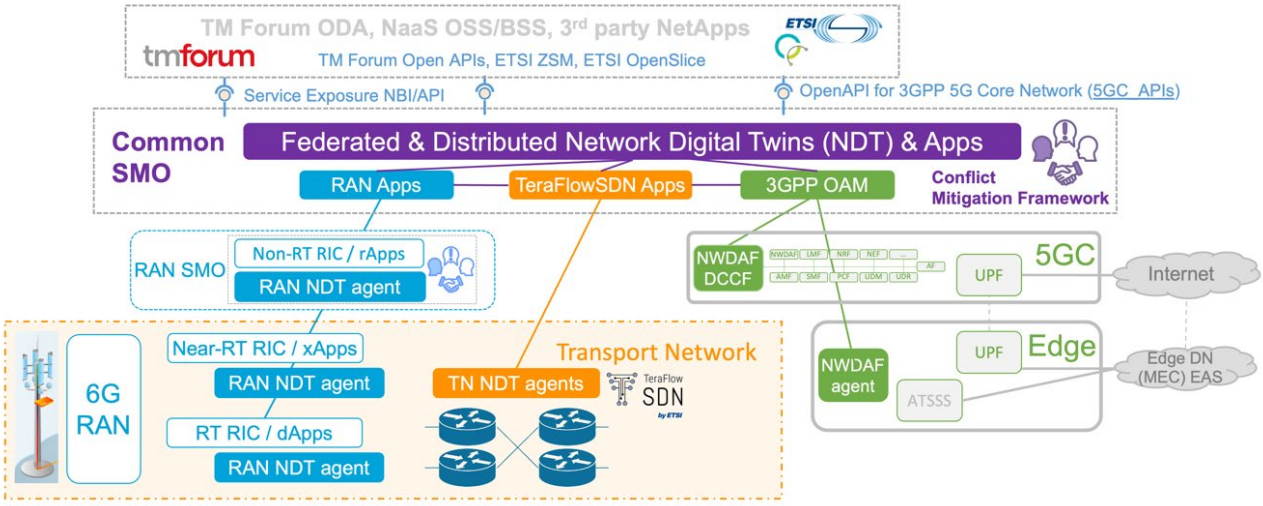

FIGURE 10: HIERARCHAL & FEDERATED SMO/AGENTS FOR 6G O-RAN 'COMMON SMO'

This design pattern of distributed analytics and intelligence, illustrated in Figure 10, supporting federated learning hierarchies is emerging as a necessity for a scalable architecture, fulfilling privacy requires (and AI legislation) of retaining potentially sensitive and massive personal data within restricted domains (using Edge intelligence and agents), instead of sending all big data analytics to a centralized data lake or store. This design pattern is also reflected within the DESIRE6G architecture, for example in Figure 8 and described in sections 2.4.2 and 2.4.3, where this need for hierarchical and federated Common SMO agents (depicted in Figure 10) maps directly to the DESIRE6G architecture of MAS with collaborating agents throughout the RAN, Core and transport domains.

3.1.2 O-RAN nGRG 'Common SMO'

Towards 6G, O-RAN is considering the evolution of these 5G capabilities within their O-RAN nGRG [16] having published a number of research reports [17].

As mentioned above, one consensual evolution towards O-RAN 6G is the need to integrate the higher SMO layers to include a system-wide cross-domain capability, including the core and transports

networks, and potentially additional Non-3GPP radios and new 6G capabilities like ISAC (integrated Sensing & Communication), RIS (Reflective Intelligent Surfaces) and others, under the umbrella of a 'Common SMO', to provide high-level 6G services like NDT (Network Digital Twins), shown in Figure 11 (from [17])

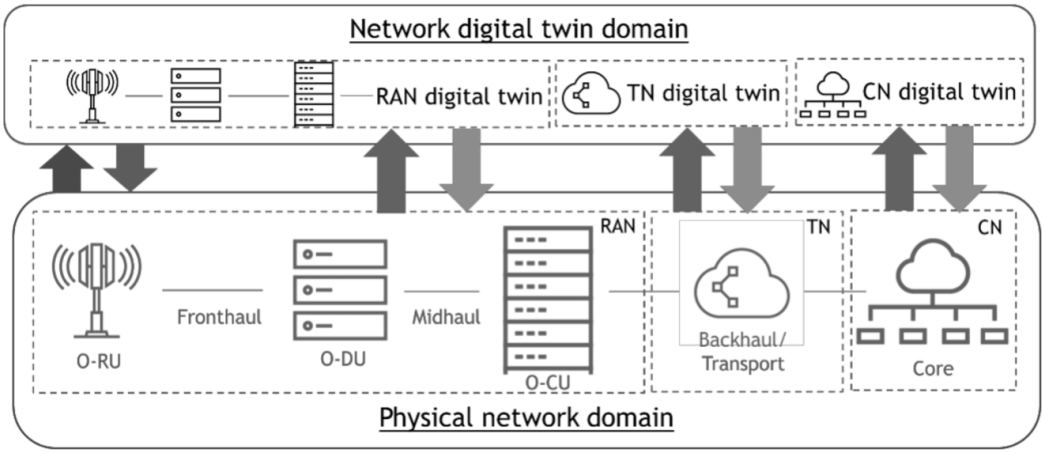

FIGURE 11: O-RAN NGRG 'COMMON SMO' FOR CROSS-DOMAIN NETWORK DIGITAL TWIN

As mentioned above, this cross-domain SMO evolution is also directly reflected within the DESIRE6G architecture, for example in Figure 8 and described in sections 2.4.2 and 2.4.3, mapping to the DESIRE6G architecture employing MAS.

3.1.3 3GPP Hierarchical Management and Orchestration

Within 3GPP TS 28.533 [18] of Release-18, there is strong trend towards exposure of service APIs, with producers/consumers, together with distribution of MDAS/MDAF (Management Data Analytics Services/Functions), between central and domain specific levels.

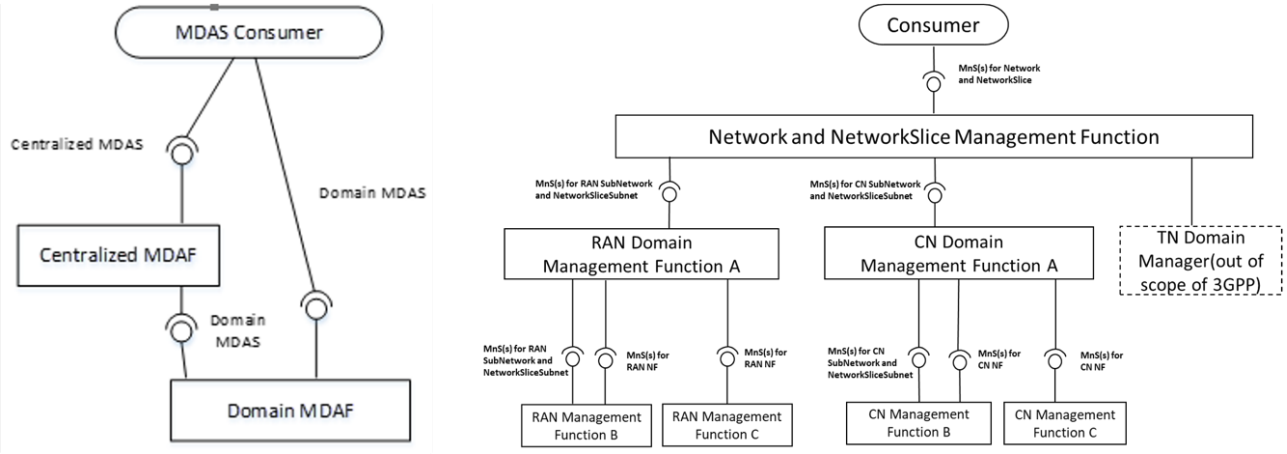

FIGURE 12: 3GPP SERVICE API EXPOSURE AND DISTRIBUTED MDAS/MDAF

Again, this same design pattern of the federation of multiple distributed domains, into a common cross-domain SMO, using exposure APIs, is repeated within the 3GPP management framework, as earlier described within O-RAN in sections 3.1.1 and 3.1.2, validating the DESIRE6G multi-layer service optimization architecture of SMO and MAS.

3.1.4 3GPP NWDAF Evolution for Federated Learning

Within 3GPP TS 23.288 of Release-18 [19] the evolution of the NWDAF function is increasingly becoming distributed between central consumer and local source analytic functions, to support flexible telemetry collection, federated learning, distributed data processing, etc, to support future 'Native AI/ML' capabilities in the RAN, with ADRF (Analytic Data Repository Function), shown below in Figure 13.

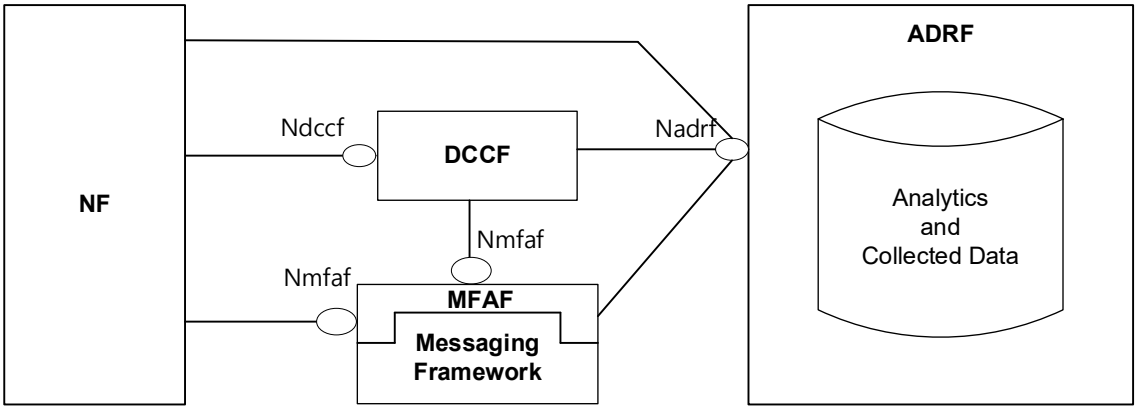

FIGURE 13: 3GPP EVOLVED NWDAF WITH DISTRIBUTED ANALYTICS

This 3GPP NWDAF distribution and federation, to support 6G 'Native AI/ML', as in the cases highlighted above for in section 3.1.2 and 3.1.3, validates that the DESIRE6G architectural design employing multi-timescale control loops, through the SMO and MAS, described in section 2.4 and illustrated in Figure 8 and Figure 10.

3.2 Service deployment and lifecycle management in DESIRE6G

Figure 14 presents the final DESIRE6G SMO architecture. The SMO acts as the mediator between the vertical deploying a service and the underlying infrastructure that implements the service. The individual components of the SMO have been described in Deliverable 3.1 [26]. In this section we elaborate on the basic functionality of the components, as well as the internal (intra-SMO) and external interactions of these components.

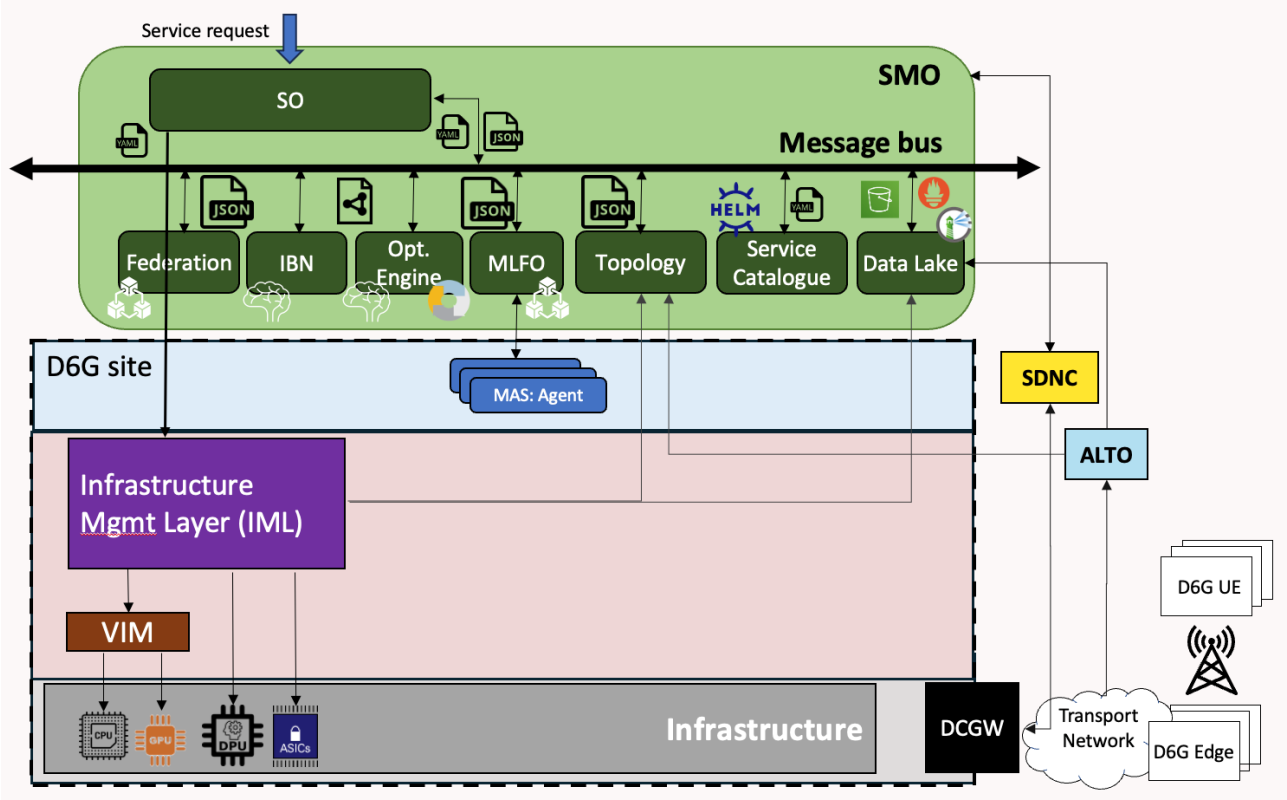

FIGURE 14: D6G SMO OVERVIEW

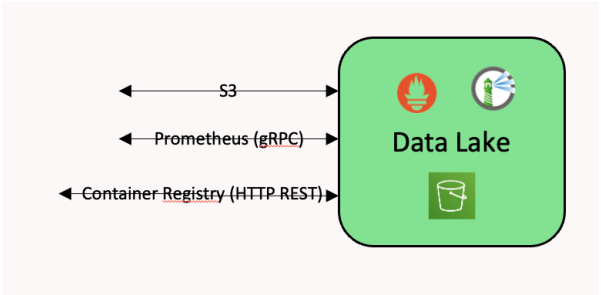

FIGURE 15: DATA LAKE INTERFACES

The Data Lake (Figure 15) is essentially a repository to store all telemetry data and binary artifacts. Specifically, it features a wrapper service that supports multiple protocols, each one of which is tailored to the specific use-case: for telemetry, it exposes a timeseries DB interface, for network function artifacts, it exposes a container registry interface, whereas for general storage it exposes an object-store interface (s3). Service graphs point to the data lake for network function- and application-level binary artifacts.

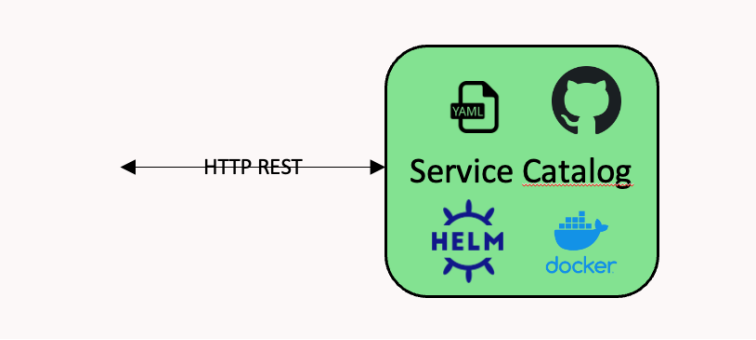

FIGURE 16: SERVICE CATALOG INTERFACE

The service catalog stores and manages the descriptors of network functions and services that are available for deployment on the target infrastructure. The catalog includes information about the VNF packages, service descriptors, and other metadata that are required for service deployment and management, such as annotated Service Graphs, application bundles etc. The service catalog component exposes a REST API with two operations: STORE and RETRIEVE (see Figure 16).

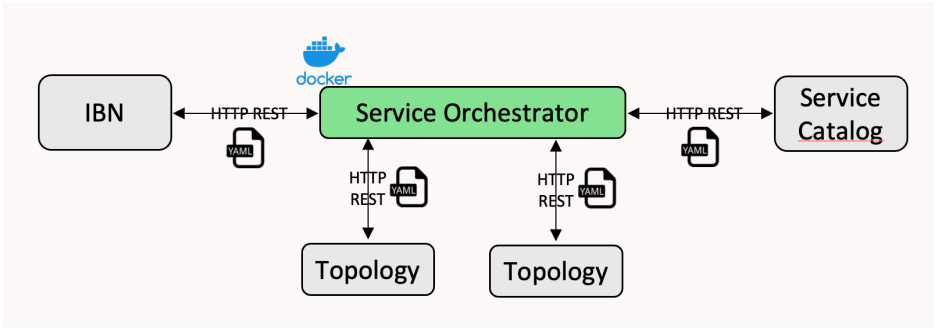

FIGURE 17: SERVICE ORCHESTRATOR INTERFACES

The Service Orchestrator is the key entity at service deployment. The orchestrator has full understanding of the operator's DESIRE6G sites via the *Topology* component that includes site resources and inter-site connections. The Topology component exposes a REST API and stores information regarding compute and network resources for each site, as well as link information between sites. The available operations for the Topology component are: ADD SITE, RETRIEVE SITE, ADD LINK, RETRIEVE LINKS (see Figure 17).

The IBN component (described in section 3.3), using information from the Service Catalog, and from the Topology component, feeds the SO with annotated Service Graphs/Descriptors that are to be deployed in the DESIRE6G sites. Essentially, the SO subscribes to a specific topic in the information bus (a message queue) and for each service description instructs the IML to deploy NF-sub-graphs to selected sites.

3.3 Intent-based service framework

Providing multiple services with different performance and functional requirements brings problems such as the complexity of network management and the need to ensure the required performance levels. Network management automation emerges as a viable solution to mitigate this complexity [22]. Concepts such as zero-touch [23] networks and intent-based networking [24] are pivotal in enabling and facilitating this automation, aiming to streamline operations and enhance efficiency.

Intent Based Network Management (IBN, see Figure 18) plays a crucial role in the realm of 6G services, where service complexity will increase significantly. At its core, IBN aligns network operations with business objectives by translating high-level intents into sub layer intents and orchestration. Intent is the formal definition of service expectations, including objectives, goals, and restrictions, for a technical system (Forum, IG1253 Intent in Autonomous Networks (Version 1.3.0), 01-Aug-2022). By focusing operations on intents rather than specific tasks, the recipient system gains significantly more latitude in selecting solution strategies and actions to implement. Embracing a requirement-centric approach is fundamental for achieving a distinct separation of concerns, fostering higher levels of autonomy within the system. IBN introduces a proactive and agile management paradigm, where networks can anticipate and respond to emerging requirements and opportunities dynamically. In the context of 6G, IBN holds a critical architectural position within the Service Management and Orchestrator framework. Positioned as a central component, IBN serves as the cognitive engine that bridges the gap between high-level business intents and the underlying network infrastructure.

In the service layer, IBN interprets and analyses incoming service requests, ensuring they align with existing services or creating new ones as needed. In this study, our focus lies in creating a new service when existing ones cannot fulfil the demand for a new service. Creating a new service is achieved through the following steps, and we will explain these steps one by one.

1. Receive business request (usually in the form of service level agreement – SLA)
2. Translate the SLA to intents
3. Creation of service description and service graph
4. Decompose the service for actual deployment (low level intents and sub service-graphs)

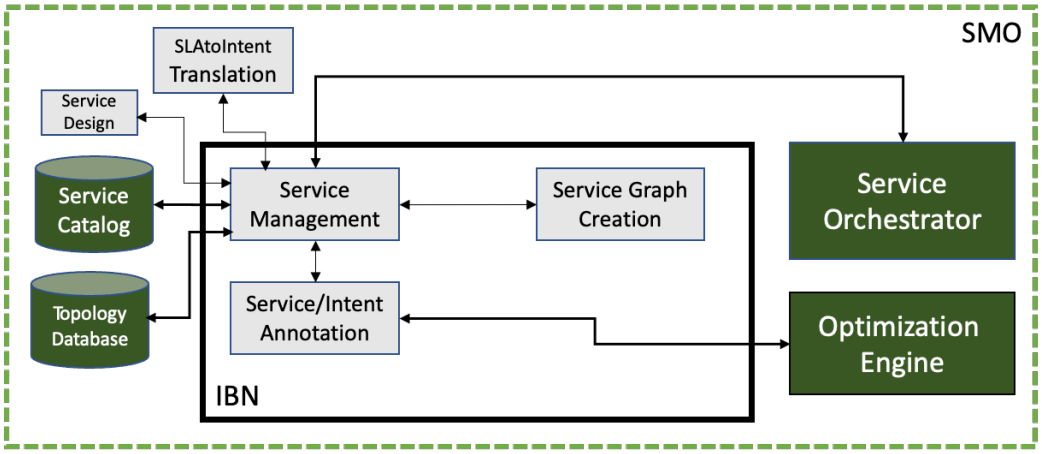

FIGURE 18: IBN ARCHITECTURE

3.3.1 Receive Business Request (SLA)

A new service request can come from various sources within the business domain or directly from customers, either in the form of unstructured or semi-structured text accompanied by SLAs. SLAs define the agreed-upon level of service between the service provider and the customer, outlining parameters such as response times, availability, and performance metrics.

SLAs necessitate comprehensive specifications to ensure effective service delivery throughout the lifecycle of network services. These specifications typically encompass various facets, including service requirements (such as required bandwidth, latency, and mean time to service recovery), service aspects (like multiple band and access point selection, and edge computing capabilities), design aspects (including availability, reconfigurability, security, and robustness), and legal considerations delineating responsibility for service degradation and corresponding compensation.

Considering the specific requests that may arise for an XR service, adherence to stringent operational standards is crucial. XR cell availability must consistently meet or exceed 99%, ensuring uninterrupted access to augmented reality applications. Users typically demand robust throughput capabilities: downlink speeds averaging 45 Mbps, with a minimum acceptable threshold of 30 Mbps, and uplink speeds averaging 6 Mbps, with a threshold of 4 Mbps during measurement intervals occurring five times per hour (38.838, 2022). Any service disruptions incur a penalty of 0.01% for future outages. Furthermore, during peak usage periods, XR UEs must meet stringent Packet Delivery Budget (PDB) requirements, ensuring that 99% of application layer packets are transmitted within 10 ms for DL and within 20 ms for UL to maintain user satisfaction. The XR cell capacity, defined by the maximum number of UEs per cell,

must ensure that at least 90% of these UEs are satisfied to maintain operational efficiency (M. Gapeyenko, 2023). These measures collectively ensure that the XR VR service meets high-performance standards demanded by users seeking immersive augmented reality experiences. Compliance to received service requirement from the business layer (SLA) is the terminal goal of the service layer.

3.3.2 Translate from SLA to Intent

The process of translating SLAs into actionable intents is crucial for seamless service provision. Firstly, it is necessary to divide the SLA into service and business domain-based segments and acquire parameters related to the service management domain. Within the SLA, the network requirements of the service must be defined structurally. Whether the SLA comes in text format or semi-structured text, important information needs to be identified with a data structure upfront.

Employing an SLA ontology is crucial as it standardizes and formalizes the representation of SLAs, ensuring clarity and interoperability across different systems and stakeholders. By using ontologies, SLAs can be queried and reasoned upon more effectively, allowing systems to dynamically assess and adapt to changing conditions. Integration with applications becomes smoother, as ontology provides a structured framework for understanding and implementing SLA requirements within various software environments. Validation and compliance checking are also streamlined, ensuring that SLAs are accurately interpreted and adhered to. Documentation benefits significantly, as ontology facilitates comprehensive and consistent documentation of SLA terms and conditions, improving transparency and accountability in service provision.

When examining standards and industry service requirements, we observe that QoS expectations are described with multiple values such as best, worse, and normal expectations, detailing how critical these are for the service. Additionally, SLAs can specify measurement periods and formulas. Furthermore, they can also indicate whether the QoS is defined for specific geographical areas or particular times or periods. Therefore, when reviewing an SLA, it is crucial to have predefined ontologies or other data structures to understand which types of information may be included and how they should be interpreted. This enables automation of the system for efficient management of QoS.

Table 1 is an example of structural representation of data in table format. It illustrates an example of non-functional requirements of a service (in this case service availability), detailing the expected levels of availability across different scenarios. At its best performance, the service is expected to achieve 99.99% availability, representing the highest standard of reliability. Conversely, the worst-case scenario

allows for a minimum of 99% availability, ensuring a baseline level that must be met even under challenging conditions. On average, the service aims for 99.9% availability, reflecting typical operational conditions. These availability metrics are not optional but mandatory, emphasizing their critical importance for service delivery. Availability is measured by averaging data over a week, indicating a continuous monitoring approach to ensure consistency in performance assessment. The geographical context specified as "Urban" suggests considerations for network dynamics and environmental factors typical of densely populated areas. Furthermore, the requirement for 24/7 availability underscores the service's commitment to uninterrupted operation, catering to continuous user needs without time-based exceptions. This table can be stored in JSON or different formats and made accessible to other components.

TABLE 1: TABULAR FORMAT FOR XR AVAILABILITY QOS

QoS Availability			
Parameters	**Best**	**Worse**	**Average**
Availability	99.99%	99%	99.9%
Mandatory/Optional	Mandatory		
Measurement Formula	Average over a week		
Geographical Region	Urban		
Time Constraint	24/7		

This communication between the SLAtoIntent Translation module and the IBN module requires an API capable of displaying negotiation, verification, and intent capability profiles. Adhering to industry standards, the TMF 921 (TMF921A, 2022) interface plays a pivotal role in facilitating this process. Essentially, the API is designed to streamline intent management processes between the two components.

3.3.3 Creation of Service Description and Graph

Upon receiving a service request, the Service Intent Management module consults the service catalog to obtain the corresponding service description. For this purpose, a POST request is made to the respective endpoint to check if a service is available in the catalog. The request body includes the name

and definition of the service. The response includes a Boolean value indicating whether the service is available or not, along with an optional message providing context about the availability. In this query, the service parameters provide specific requirements (e.g., KPIs, constraints, cost, level of isolation) for the service being checked. These parameters allow for a more detailed assessment of the suitability of a service description based on its specific characteristics and requirements. If the service is not listed, the orchestrator initiates the creation process. Service intent management seamlessly collaborates with service design, jumpstarting the design process for the new service. Through this interface, essential details such as service name, ID, information, and a basic service graph are obtained, setting the stage for further steps. Once the new service ID and names are obtained from service orchestrator, relevant services are retrieved from the service catalog. This interface not only incorporates the service scope, features, and sub-services into the API but also shares critical information regarding related services, encompassing sub-slices, VNFs and PNFs facilitated by YAML files. In addition to catalog information, topology data plays a pivotal role in service provisioning.

The service graph definition typically outlines the arrangement and interconnection of various VNFs and PNFs within a network service. It contains the definition of the VNF forwarding graph(s). Each component has associated properties specific to its type. For instance, a request for a service in a city centre necessitates the retrieval of topology connections specific to that area, contingent upon the selection of the Access and Mobility Management Function (AMF) and Session Management Function (SMF) group. Selecting the right SMF and AMF is crucial for service graph creation with service geography restriction. By choosing these functions strategically, organizations can ensure regulatory compliance, optimize performance, and tailor services to meet the unique needs of users in the targeted region. This approach enhances service delivery, fosters customer satisfaction, and mitigates legal risks associated with regional regulations.

3.3.4 Decompose the Service for Actual Deployment

Intent service management serves as the nexus for consolidating topology and catalogue information, channelling it to the Service Graph Creation module, which intricately weaves together the service graph. Intent and service graph are transmitted to the Service and Intent Annotation module, where they undergo decomposition into sub-services, intents, and sub-graphs - utilizing the Optimization Engine, for transmission to subdomains (via the Service Management and the Service Orchestrator). The Service and Intent Annotation module ensures that the high-level service requirements and objectives are translated into concrete sublayer intents and sub-graphs at different network domains, enabling

efficient service delivery and resource optimization across the entire network infrastructure. This intricate orchestration is supported by the Optimization Engine, enabling the selection of the D6G sites and inter-site traffic paths for service deployment. Furthermore, domain-based risk assessment ensures the robustness of service delivery. Following, the Service Orchestrater is sending the respective partial service requests to the underlying technology/administrative domains, facilitating service deployment based on informed decisions.

Once the service is successfully instantiated, the orchestrator notifies IBN, triggering the dissemination of pertinent information regarding the new service's status to the service catalog, culminating in the addition of a comprehensive service record.

3.4 Optimization Engine

The Optimization Engine (OE) is a module that operates at the SMO-level. The main objective of the OE is to supervise and coordinate AI/ML operations for the automation of service life-cycle management. For example, during service deployment, the OE receives network service requests in the form of service graphs, annotated with their respective functional (HW requirements, etc.) and non-functional (E2E latency, availability etc.) requirements. With the use of advanced AI/ML models the OE partitions the service graph, decomposes its non-functional requirements and matches its functional requirements in an optimized way, as seen in Figure 19.

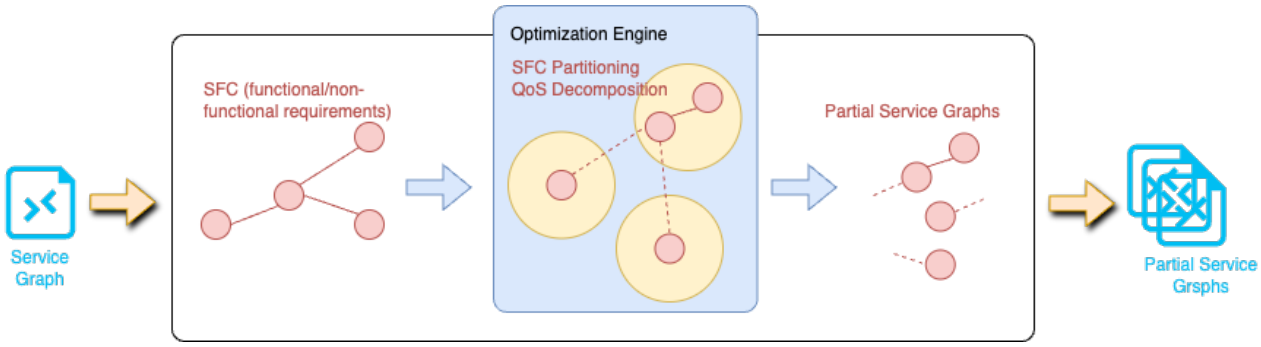

FIGURE 19: OPTIMIZATION ENGINE SERVICE PARTITIONING AUTOMATION.

All endpoints and sub-modules of the OE are shown in Figure 20. Specifically, the OE receives the annotated service requests from the IBN in its respective endpoint. The incoming service graph is partitioned by the selected AI/ML model, based on the respective non-functional/functional requirements included in the service request and the current substrate information (including topology

and available monitoring information denote as network state) that are retrieved from the Data Lake via the DESIRE6G system bus. The final partitioned services are encoded again into a JSON file and returned to the IBN. An additional Management API endpoint is used to remotely monitor and control the OE module

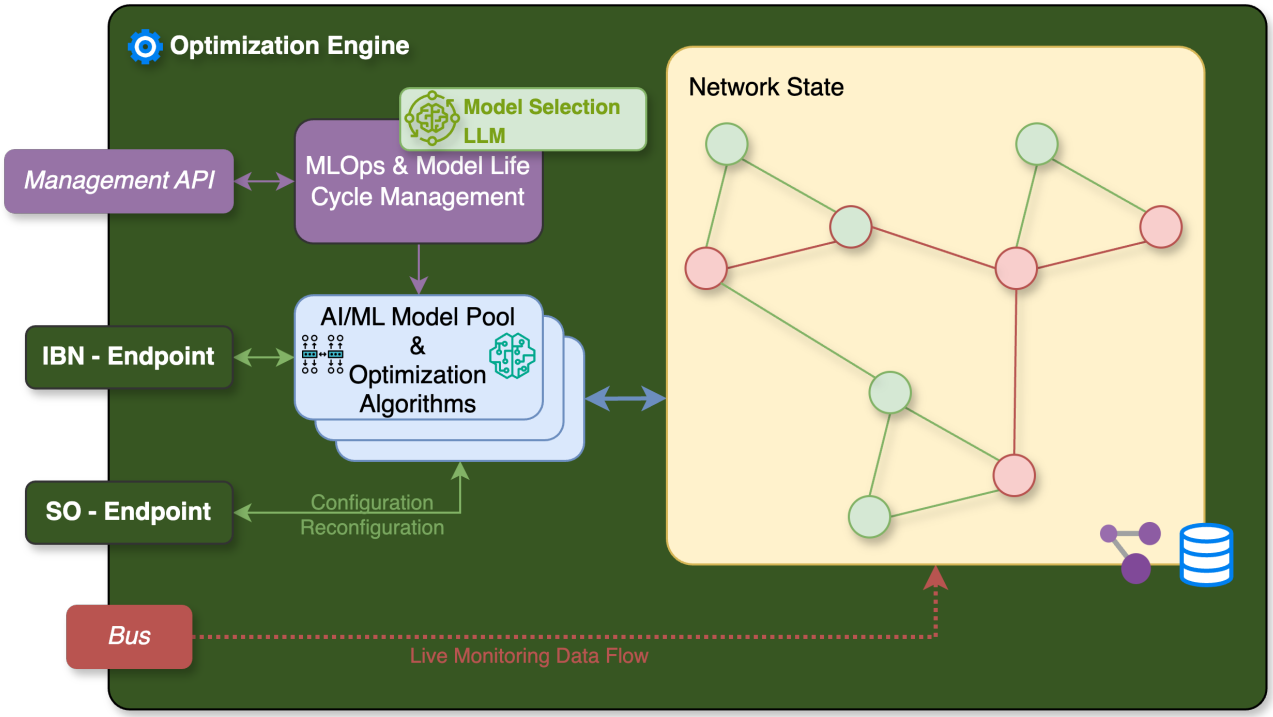

FIGURE 20: OPTIMIZATION ENGINE MODULE INTERNAL ARCHITECTURE.

The workflow of the OE in this example, as depicted in Figure 21, consists of the following tasks:

(1) receiving the service graph from the IBN module,

(2) auto-selecting the AI/ML model from the Optimization Model Pool based on the SLA of the incoming request from the IBN,

(3) Partitioning the service graph by decomposing its respective QoS requirements through running the selected AI/ML model,

(4) encoding and sending the generated service to the IBN.

The models are continuously evaluated and trained based on the feedback received via the monitoring system (network state data in Figure 21). The workflow can be monitored and controlled via the Management API and its client application.

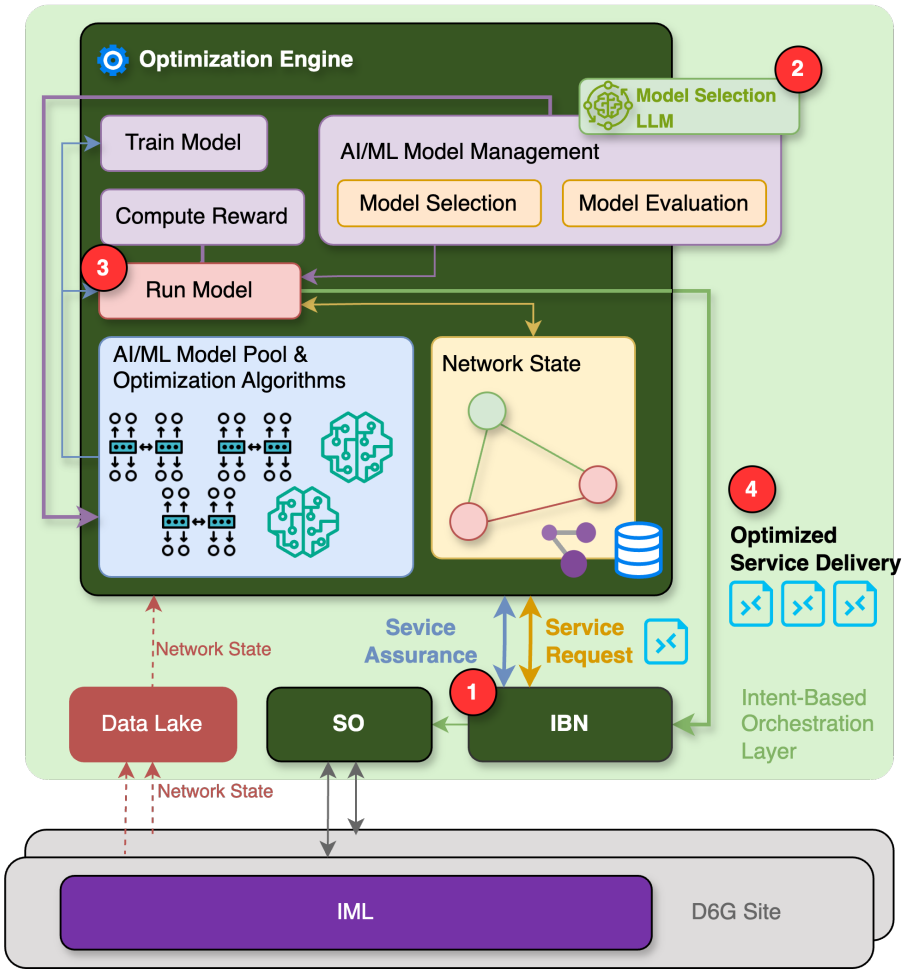

FIGURE 21: OPTIMIZATION ENGINE AI/ML WORKFLOW.

The OE holds an integral role in automating the service deployment and optimization. It is used by the IBN and the SO modules to optimize service life-cycle management processes.

4. Service optimization and assurance

4.1 MAS and MLFO

This section is devoted to the two main D6G subsystems (sketched in Figure 22), that enable intelligent near-real time control of the different network functions that compose a service graph:

- The **MAS** implements distributed network intelligence closer to the physical infrastructure, as it is responsible for receiving service-specific monitoring information and fine-tuning the network and compute resources. It configures and uses the D6G pervasive monitoring system to receive service-specific performance indicators, e.g., end-to-end latency for latency-sensitive or latency-critical services. The MAS can apply AI algorithms for optimization purposes and can have several roles based on its actual purpose. The different agents share information with each other and this way they can solve complex, network-wide optimization or fault handling functionality. This section will show main functionalities, requirements, and capabilities of MAS using some illustrative examples.
- The **MLFO** is responsible of deploying and, if needed, reconfiguring the MAS of a given service. Running in the centralized SMO, it is in charge of creating AI/ML pipelines and relating them to the target service. AI/ML pipelines are associated to network entities and need to be deployed and reconfigured properly according to needs, e.g., flow rerouting of a service requires moving agents (with their performance data and models) among different D6G sites. The way how MLFO executes MAS deployment and attachment during service deployment will be specified later in this document, in the message flows section.

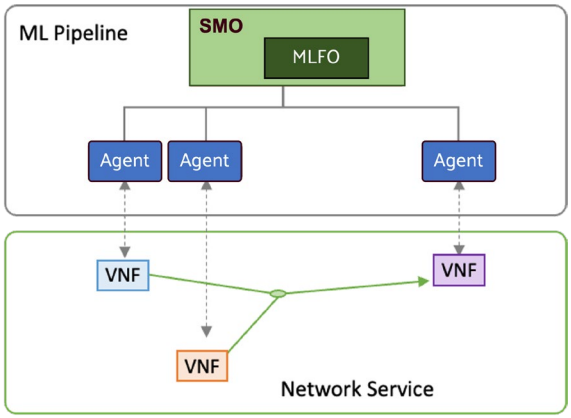

FIGURE 22: MAS AND MLFO SCHEME

A general overview of MAS is provided in Figure 23. Agents are typically designed to interact with different D6G subsystems and components. In particular, MAS needs to be able to interact with the main following components:

- Local / domain-specific controllers both for monitoring data and for control:
  - Near Real-Time (NRT)-RIC: for RAN telemetry processing and resource/policy control.
  - SDN controller and/or the site routers of the D6G sites for transport-related telemetry processing and reconfiguration.
  - IML for local resources telemetry processing and reconfiguration, e.g., scaling.
- In-network telemetry data producers.
- SMO, to provide data, models, and control actions, and receive guidelines and policies. These policies are provided at service deployment time and can be updated during operation, according to the decision of global view algorithms in charge of actions such as MAS reconfiguration, as well as supervising and tuning MAS operation.

Moreover, agents within a MAS communicate among them to exchange telemetry measurements, notifications, and models. This is required to perform intelligent service control actions that are beyond the scope of a single agent, e.g., actions to be performed in an agent that is in charge of a given D6G component (e.g., change the routing policy in a given router of a D6G site) requires collecting and analysing telemetry data collected along the whole e2e service graph to deal with the proper action.

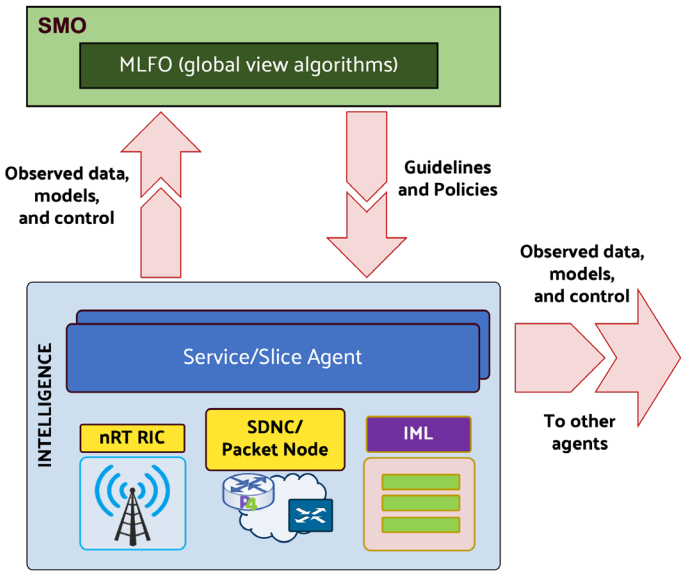

FIGURE 23: MAS OVERVIEW

To illustrate the abovementioned concepts, let us focus on a use case that combines different solutions for the near real-time control of e2e 6G services. Figure 24 presents the scenario, where a service provides connectivity between an unmanned aerial vehicle (UAV) and an application function (AF) that runs real-time augmented/virtual reality (AR/VR) high quality video reconstruction. Strict e2e and segment latency and jitter requirements, imposed by the application, require continuous monitoring of a large number of stream flows, to dynamically make routing and edge computing resource selections that satisfy the committed performance.

Assuming service pervasive monitoring capabilities (explained in detail in the following section), the UAV adds geo-location coordinates as metadata to the telemetry packets. Moreover, the RAN aggregates signal quality indicators, e.g., channel quality indicator (CQI) and the latency experienced in the access segment. Finally, the intra/inter-switch latencies and jitter are collected from the packet nodes. The different agents in the MAS collect that data and perform analysis in order to find the best routing policy to be applied in packet node NS1.

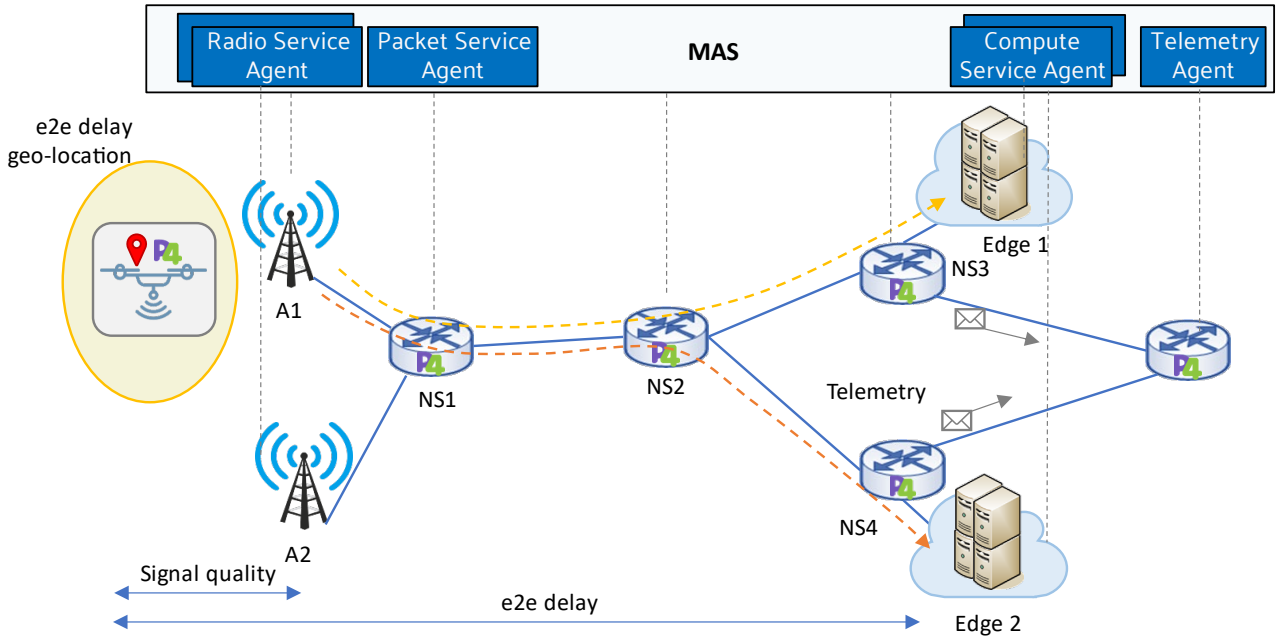

FIGURE 24: ILLUSTRATIVE USE CASE

The detailed architecture of each of the agents, as well as the relationship between them is illustrated in Figure 25.

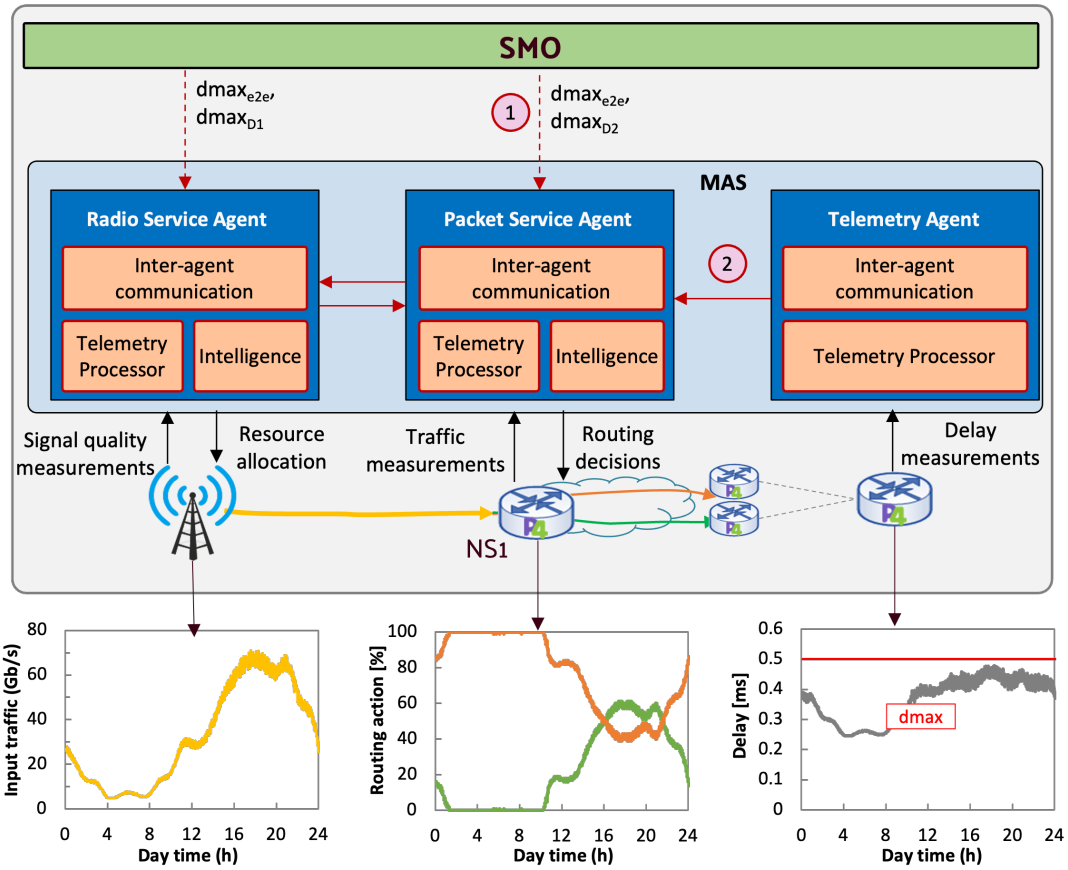

FIGURE 25: INTELLIGENT ROUTING USE CASE

Service agents might include:

i) a telemetry processor to collect from the local node and process telemetry data, including intelligent data aggregation;

ii) an inter-agent communication module, which is used for telemetry data distribution and state and model sharing; and

iii) technology-specific (i.e., RAN, packet, etc.) intelligence for autonomous decision making based on local and remote observations.

As already anticipated, the SMO provides guidelines to the MAS, while giving freedom to the MAS for operating on the resources allocated to the service. Thus, the agents receive from the SMO the resources that can be used for the connectivity service and the max e2e delay ($dmax_{e2e}$) that needs to be ensured (labelled 1 in Figure 25). Based on the required performance, let us assume that each segment is assigned with an initial budget delay $dmax_{Di}$ (according to the specifications in intent-based service framework section in this document).

Once in operation, e2e delay and other performance indicators are measured, and results distributed among the agents in the MAS (2). For instance, in the case of the radio segment, measurements and forecasted metrics are used to enforce the resource block group adaptation with margin in time, reducing the service SLA violations. However, imagine that at some point in time, the radio segment cannot provide the committed delay in its domain. In this case, the RAN agent announces the new delay budget for its segment to the other service agents, so packet agents can make decisions, e.g., changing routing, to ensure the new budget delay in their domains.

The illustrative graphs in Figure 25 show how an increase in UE traffic (bottom left) is generating high congestion and resource utilization in the route that is being used (orange path connecting to edge site 1), which can be observed by an increase in the e2e delay (bottom right). To avoid exceeding the target maximum delay, routing actions are performed at the switch NS1 (bottom middle), so that load balancing is performed by routing part of the traffic through green path towards edge site 2).

During the service assurance phase, in the case that domain policies prove inadequate for resolution, MAS escalates the problem to the SMO; it notifies the IBN through the SO that the sub-intent cannot be fulfilled, requesting a solution. This step involves evaluating options such as creating new sub-graphs, prioritizing services, and implementing service-specific solutions. If a viable solution is identified, sub-intents and subservice graphs are updated and distributed to relevant sub-domains. Should a solution not be feasible, customers are informed that the requested service cannot be provided, and communication proceeds according to SLA guidelines outlining further steps.

In section 6 (message flows), the interaction of MAS with other D6G components will be shown.

4.2 Service monitoring

The success of forthcoming 6G systems hinges on two critical factors: pervasive network intelligence for robust network automation and efficient network control to meet the demanding performance criteria of 6G services. To achieve this, there's a pressing need for pervasive, real-time monitoring systems that complement pervasive intelligence, automated management, and near-real-time control. However, current monitoring practices lack a comprehensive framework that offers global visibility across all resources, hindering scalable network management and control across different timeframes.

The DESIRE6G system addresses this gap by embracing the concept of pervasive data-driven monitoring, which underpins zero-touch control, management, and orchestration. Monitoring data can

stem from either service elements, aligning with DESIRE6G's definition of end-to-end service, or the infrastructure itself.

In DESIRE6G, the **Service Monitoring** focuses on assessing the performance and availability of specific services or applications critical for business operations. This involves collecting data with high granularity to facilitate closed-loop operations that uphold Service Level Agreements. In addition, **Infrastructure Monitoring** involves tracking the health and performance of the underlying hardware, software, and network components supporting these services. This encompasses a range of metrics covering servers, storage, networking equipment, databases, and other essential infrastructure elements.

4.2.1 Reference Scenario

Figure 26 shows a simplified reference scenario including a multi-site DESIRE6G environment. More specifically, the scenario involves the User Equipment (UE), attached to the RU at DESIRE6G site 1, communicating with a ServiceApp, running in a different premise, e.g., at DESIRE6G site 2. Possible applications of the proposed scenario are the DESIRE6G use cases, to consider as service apps an AR/VR app or a Robot Digital Twin app.

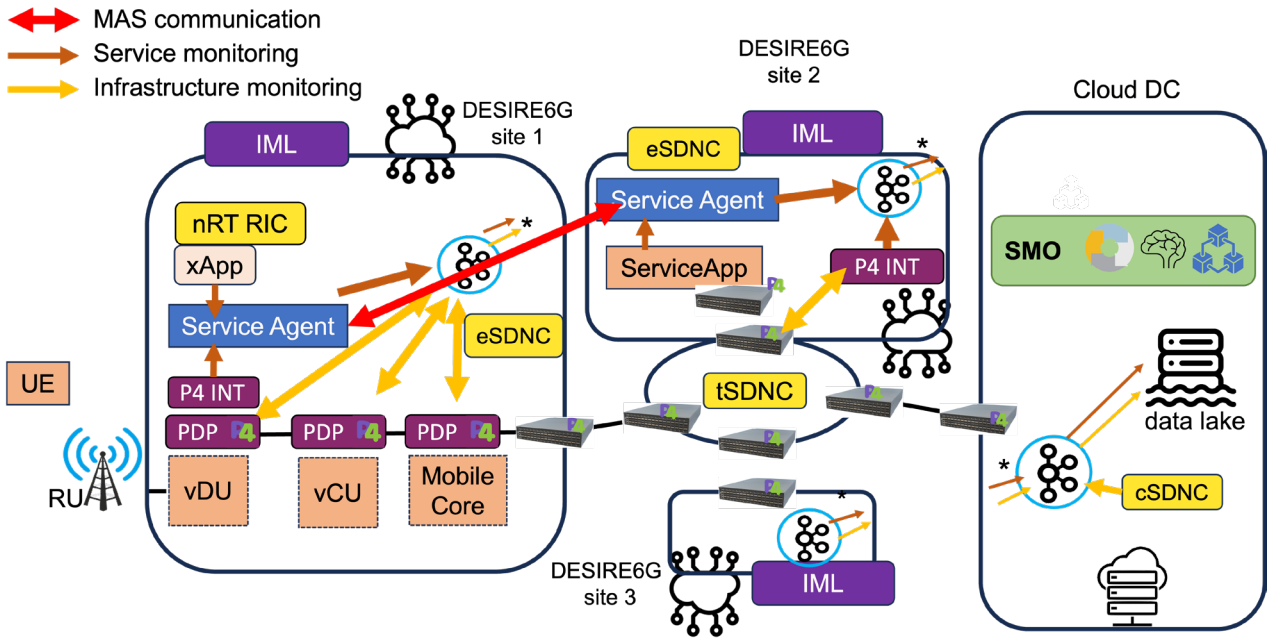

FIGURE 26: PERVASIVE MONITORING ARCHITECTURE SCENARIO

Cloud DC: The overall life-cycle management of the active network services is controlled by the DESIRE6G SMO running e.g., in the central cloud DC of the MVNO or the infrastructure provider. The SMO is responsible for the end-to-end deployment and life-cycle management of a network slice, where DESIRE6G network services run. It interacts with the local components (i.e. the IML) at the DESIRE6G sites for the deployment of services, pods and applications as well as handling the SDN transport network through the tSDNC.

In D6G SMO, components such as SO, IBN and the Optimization Engine, encompass different component/functionalities to control, manage and monitor the infrastructure across the different technology domains throughout different phases of D6G service lifecycle management (e.g., service deployment, service assurance). The full details on the 1st version of the SMO architecture and its functional modules are reported in Deliverable 3.1 [26].

Transport Network: An instance of an SDN controller (cSDNC) is responsible for intra-DC networking. It also acts as parent SDN controller for the end-to-end transport configuration, leveraging the hierarchical paradigm, with child SDN controller instances allocated at the edge nodes (intra edge networking) and the transport network. The cloud DC connects to the remote edge nodes by means of a transport network, representing a metro scenario. All connections are established using P4-programmable and legacy switches deployed across the metro network. The topology in Figure 26 is a simplified example. The transport network is controlled by a transport SDN controller (tSDNC) allowing flexibility and high data-rate for the communication among the different sites.

DESIRE6G sites: Three DESIRE6G sites are shown in the figure. Each one is controlled by a dedicated IML that is responsible for (local) deployment of the network service employing infrastructure cloud entities (i.e., K8s pods), exploiting available accelerators, and the network configuration.

DESIRE6G site 1 provides radio connectivity. More specifically, the 3GPP NR 7.2 is considered, where the Radio Unit (RU) is connected to the virtualized Distributed Unit (vDU). The vDU is then connected to the virtualized Central Unit (vCU). The considered scenario considers a latency-sensitive application that requires the deployment of the Mobile Core function at the same edge node, implementing the complete 3GPP stack. For other network slices with no latency constraints, the core function can be potentially deployed in other DESIRE6G sites across the edge-cloud continuum or even at the cloud DC.

The adoption of a programmable data plane opens the way to leverage hardware acceleration for some of the network function components (i.e., the User Plane Function – UPF or the CU-UP). An instance of

near real-time RIC is available at the edge enabling the support for near-real time closed loop operations, along with the respective service agent, the functionality of which is implemented as an xApp.

Pervasive monitoring components: Figure 26 shows also the crucial components of the pervasive monitoring framework and their relationship.

At the each DESIRE6G site, monitoring can be performed at the level of the service (represented in red) or the level of the infrastructure (represented in yellow). In the example, three types of service metrics can be collected: (i) metrics form the radio segment (i.e., the statistics of the slice or of the UE), by using the respective xApp, (ii) the metrics collected from the PDP devices, collecting telemetry information (per-packet metrics during packet transmission) from the data plane and (iii) the metrics related to cloud elements of the service (i.e., the ServiceApp at DESIRE6G site 2).

In addition, relevant metrics can be collected from the infrastructure entities, in example the network devices (nodes and link) and the cloud resources. All those metrics are collected by specific frameworks/probes and are made available for the data consumers (i.e., AI/ML-based routines).

To support these high-performance data pipelines at the infrastructure level, we employ Apache Kafka [27] as the underlying message broker. Apache Kafka is an open-source distributed event streaming platform originally developed by LinkedIn and later donated to the Apache Software Foundation. It is designed for handling real-time data feeds and stream processing. All data is labelled, allowing data filtering and data aggregation. Apache Kafka is a robust and highly scalable platform that plays a critical role in modern data architectures, enabling real-time data streaming, event processing, and data integration for a wide range of applications, from log aggregation to real-time analytics and beyond.

Therefore, in DESIRE6G both service and infrastructure metrics are pushed to the local Kafka-based cluster. In fact, each DESIRE6G site is equipped with a Kafka cluster, being able to act as metrics re-distribution system towards the entities responsible for intelligent decision-making pertaining to service assurance.

Considering the service metrics related to the considered example, at DESIRE6G site 1, the system collects service metrics from the radio segment and PDP resources. While, at DESIRE6G site 2, the P4 based metrics and the cloud metrics, related to the ServiceApp are collected. DESIRE6G site 3 is not involved in the service and so no service-related metrics are active.

4.2.2 Monitoring components interaction

In Figure 27 an overview of the monitoring components and their relationship is highlighted. On the bottom of the figure examples of heterogeneous resources are depicted, considering wired and wireless networking elements and cloud resources. Once a new service is enabled, a service agent is deployed in each DESIRE6G site supporting the service, by the MLFO of the SMO. This agent is responsible for triggering the activation of the monitoring elements. The communication with the different monitoring elements is performed using a publish-subscribe system. In this system different topics can be adopted for slicing the messages. In the figure, the configuration topic is highlighted with yellow.

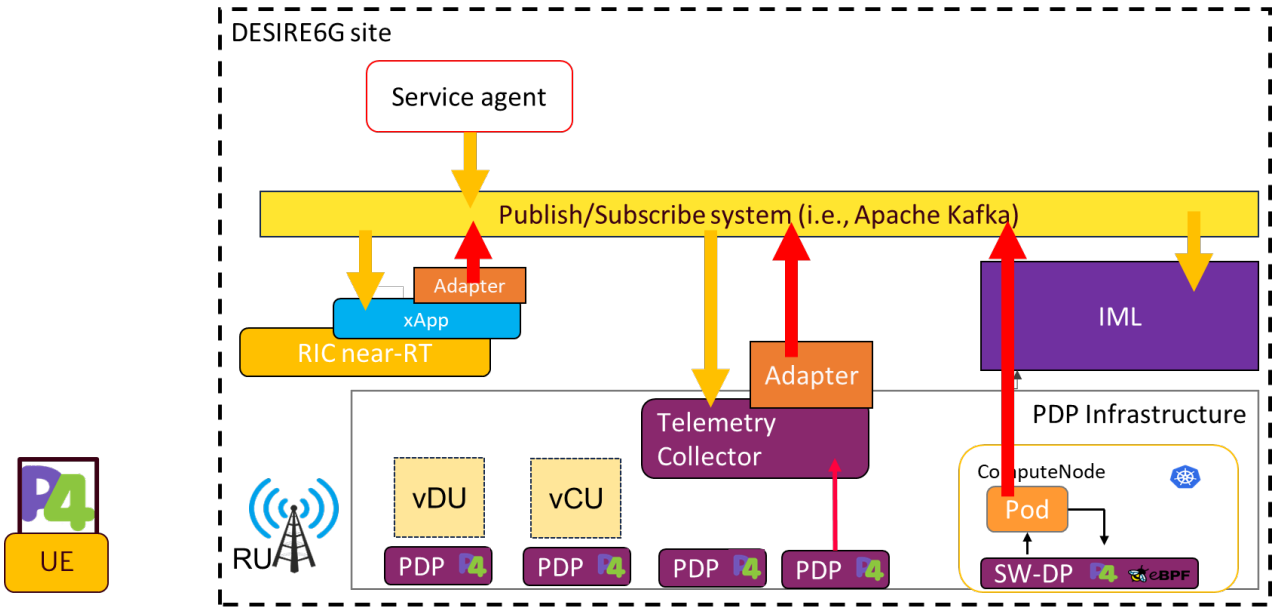

FIGURE 27: INTERACTION AMONG MONTORING COMPONENTS

In this case, the service agent is acting as producer, generating messages to inform all the subscribed monitoring components of the service details to be used for the metric collection. This information includes a metric list, tagging information and communication details (IP addresses and/or ports). Different functional elements (i.e., the IML, the RIC) are subscribed to this topic, acting as consumers, and will react to the messages enabling the generation of monitoring messages. Possible examples of reaction are (i) the activation of a new thread at the Telemetry Collector to generate data related to a new service, or (ii) the activation of an xApp at the nRT RIC. Then, the collected monitoring metrics will be sent to the service agent, leveraging the publish-subscribe system, by adopting a dedicated topic

(represented in red in the figure). In this case, the service agent acts as collector, to retrieve all the relevant metrics to perform the closed loop operations.

4.2.3 In-Band Telemetry Case Studies

This section summarizes the activities considered and developed in the context of DESIRE6G regarding the adoption and the extensions of standard telemetry approaches. As shown in Figure 28a, telemetry information is collected during the transmission of the packet and then appended to it (in-band), to be transferred to the egress switch. Here, the headers are combined in a single INT Report towards the collector. This INT Report includes all the data provided by all the traversed switches, making the Telemetry Collector aware of the condition of all the devices.

INT allows the collection of fine-grained network information in real-time, increasing network visibility, at the cost of network overhead e.g., high bandwidth consumption or decreased packet processing speed due to the additional telemetry related information. Several INT approaches have been recently proposed that attempt to alleviate the transmission overhead of INT, while maintaining a high degree of monitoring accuracy, e.g., by exercising per-flow aggregation (PFA) that is spreading telemetry values across the packets of a flow. The trade-off between monitoring accuracy and INT overhead has been thoroughly investigated in literature [47]. We employ standard INT approaches [46], such as “Postcard” or INT-MD mode, along with lightweight PFA INT approaches, namely DLINT and PLINT [47]. in the context of DESIRE6G monitoring scenarios. More information on the employed INT approaches is available in Deliverable 4.1 (Section 5.3.1) [42].

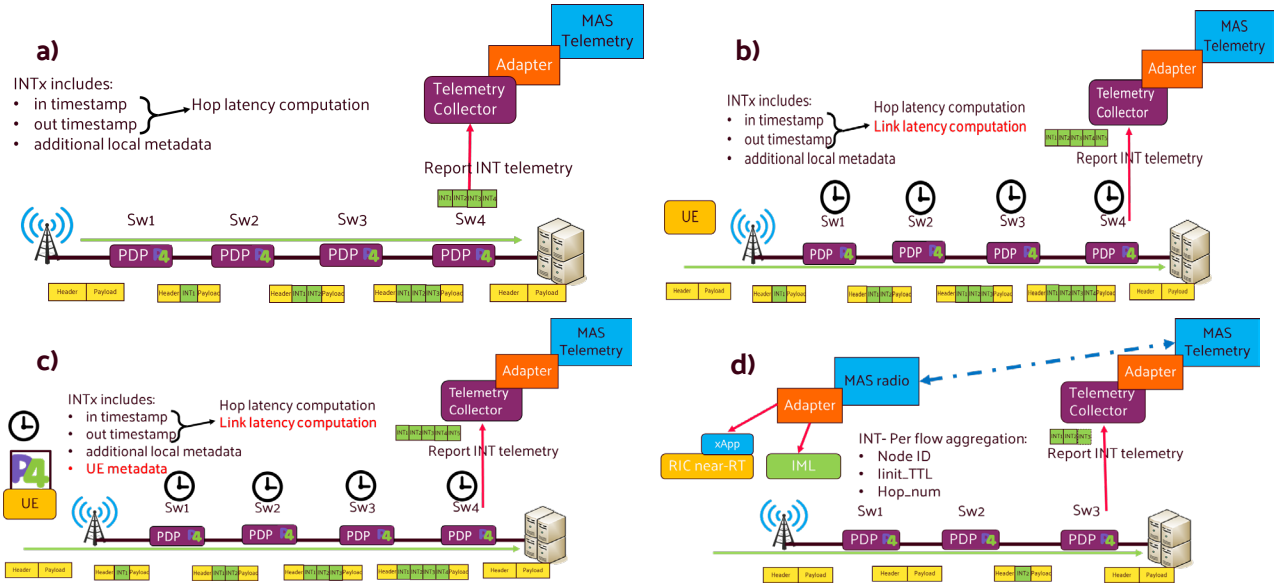

FIGURE 28: PERVASIVE TELEMETRY CASES OF STUDY

E2E Delay with Deep Clock Synchronization: In Figure 28b an enhanced INT scenario is presented, where deep clock synchronization is exploited in the network switches. In the proposed scenario, the timestamps of all the switches will be in synch, allowing the evaluation not only of the hop latency, but also of the link latency and the overall end-to-end latency, by processing the timestamps provided by the different switches.

E2E delay with Deep Clock Synchronization and PDP at the UE: A possible evolution of the system is shown in Figure 28c, including in the telemetry INT headers also the UE metadata. This can be achieved by employing data plane programmability at the UE side to produce UE metrics. In this case the INT header can include ingress timestamp, egress timestamp, additional local metadata, UE signal status/quality, and application specific metrics. These additional metrics can be then adopted to perform traffic steering decisions and/or reconfigure the other segments of the network. As described in the previous scenario, the same concepts can be applied here, being able both to measure the hop latency and, by exploiting the clock synchronization, to evaluate the E2E link latency, including the radio link between the UE and the base station.

In the considered heterogeneous scenario (e.g., RAN and PDP segments), the pervasive monitoring can be applied in different ways. In the programmable segment, the data collection is performed as described in the previous examples, with the adoption of a Telemetry Collector to feed the data to the service agent. In the radio segment, the data retrieval is performed via xApp, feeding the data to the

service agent. In fact, by leveraging the nRT RIC, a specific xApp can be adopted to collect the telemetry data of interest from the slices (per-UE or aggregated per-slice).

Path Tracing: The adoption of the pervasive telemetry approach can be applied for the path tracing evaluation, a metric useful to support mobility and service assurance in the envisioned DESIRE6G architecture. This metric helps to track the route that data packets take in the network. Figure 28d presents the path-tracing scenario, including the telemetry INT headers. In the example, the INT header can include the *switch_id* or the *switch_id, init_TTL* and *hop_number* for the case of DLINT and PLINT respectively. As it will be further explained in Section 7, the DESIRE6 infrastructure functions fully or partially take over the roles of some 5G functions like UPF. For example, user mapping to network services and the routing to the appropriate cell is solved by the infrastructure (see section 7.2 for details). According to this approach, path tracing can ensure the signalling mechanism in DESIRE6G needed for, e.g., mobility support. Note that this type of handover is particularly beneficial for strictly controlled end-to-end services, where the data paths need to be predefined and potentially resource-reserved. For less stringent services, a more automatic solution (see section 6.7) is usually preferable and should be considered the default option.

Telemetry collector: Figure 29 shows the DESIRE6G two stage Telemetry collector. Network switches generate INT headers that carry the switches' telemetry data, which may be attached to every data packet or only to certain tracked flows. Each packet header includes switch metadata, such as timestamps, queue length, and interfaces. The aggregation switch receives these headers, extracts the metadata, and stores it in a memory buffer. When the buffer is full, the aggregation switch constructs a new packet containing the aggregated metadata and forwards it to the Telemetry Server.

As described in Figure 29 this is the information needed to reconstruct the path on for each approach. Path updates (e.g., due to handover or service degradation) at the 1st level telemetry collector (e.g., INT collector) can be used by the DESIRE6G multi-agent system in place to update the corresponding forwarding rules at the respective routing NFs.

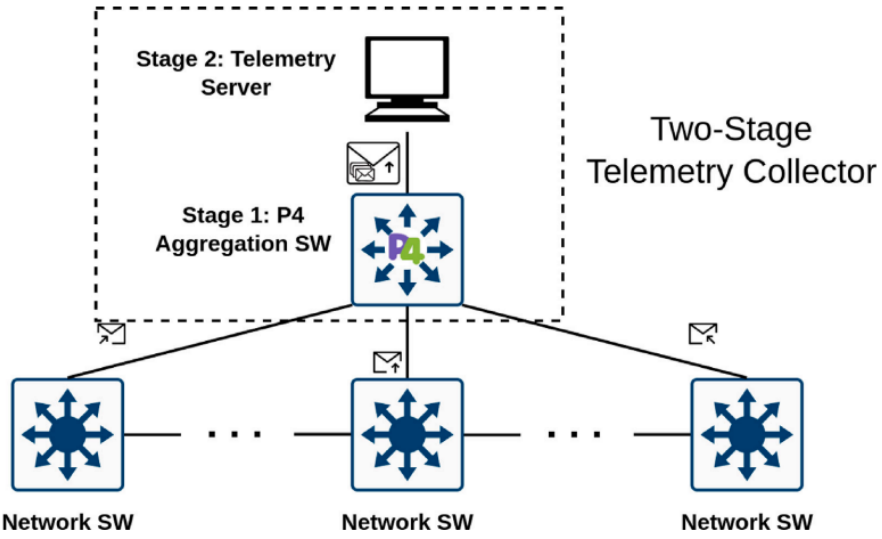

FIGURE 29: TWO STAGE TELEMETRY COLLECTOR USED FOR POSTCARD TELEMETRY IMPLEMENTATION

5. Cloud native data plane with IML

In the DESIRE6G architecture, the data plane resources available at a D6G site are managed by the Infrastructure Management Layer (IML). We follow a similar way to cloud native application deployment, e.g., in a Kubernetes cluster where proxy load balancers ensure seamless load distribution among instances and handles up and down scaling. This approach ensures high-level of flexibility in using computational resources. In DESIRE6G, we adopt the cloud native idea by realizing three key data plane concepts: (i) hiding the implementation details of the underlying network function data planes, (ii) providing seamless offloading and optimization in the data plane, i.e., hardware target selection, disaggregation, (iii) providing tools to better use the resources of hardware data planes with network function isolation, e.g., by enabling the deployment of multiple network function data planes to the same hardware.

5.1 Network function model

The network service in DESIRE6G is described as a forwarding graph employing a set of network functions. Though this representation of the service focuses on the packet-level data flow, each network function is composed of a data and a control plane entity. The data plane entity is responsible for processing packets of data flows while control plane implements the high-level business logic of the network function. This component is responsible for configuring and managing the data plane behaviour by inserting/removing rules into lookup tables or reading counters/registers, etc. The data plane operates at packet processing timescale (nanoseconds to microseconds) while the control plane is much slower (~10 milliseconds to seconds). The proposed model is depicted in Figure 30.

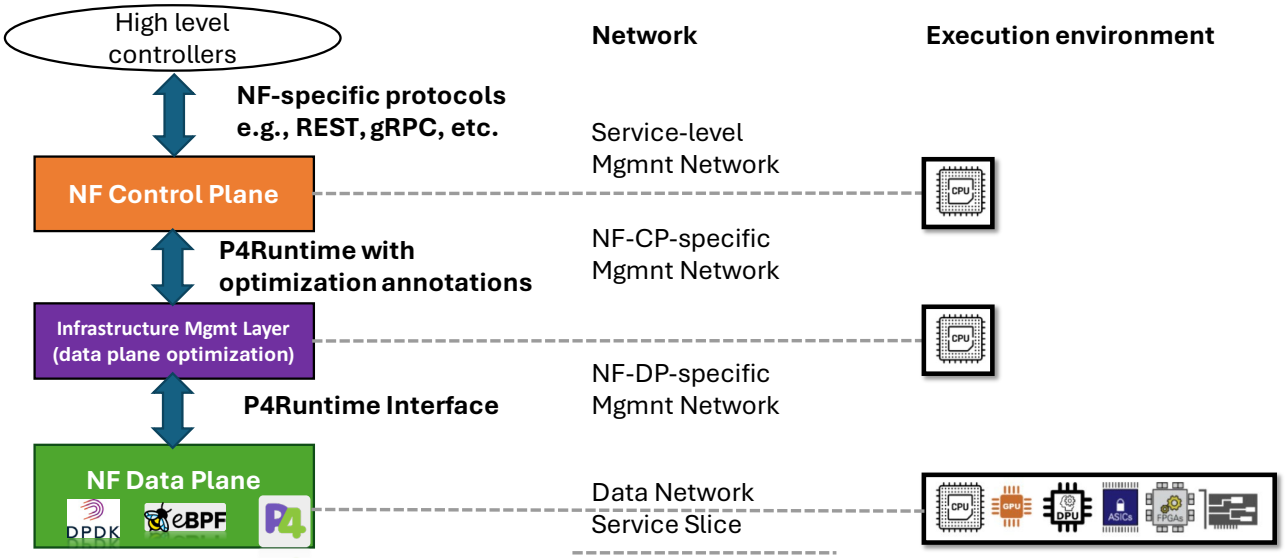

FIGURE 30: INTERACTION OF NETWORK FUNCTION CONTROL AND DATA PLANES VIA IML

An NF data plane component implements the packet processing logic and can be executed on various targets including smartNICs, ASICs, FPGAs, IPUs and DPUs, in addition to traditional CPU resources. We assume that each NF data plane has one or more L2 interfaces connected to the data network. Each interface can be used as input or output interface or both. At packet arrival the input interface can also be considered to identify the packet processing behaviour of the NF (e.g., if packet arrives on interface 1, a set of ACL rules is applied, while for interface 2 another set of rules is executed.), while the selected output interface is considered in NF routing to determine the next network function in the forwarding graph. Note that both input and output ports can potentially carry extra state information in addition to the packet content.

In addition to the interfaces connected to the data network, an NF data plane instance is also connected to a NF-DP specific management network, representing a virtual link between the NF-DP instance and the IML. The communication channel between the NF data plane and IML relies on the standard P4Runtime protocol over gRPC. P4Runtime is a generic API to setup, control the objects in the data plane. Note that - despite its name - P4Runtime can also work with non-P4 data planes. It also means that each NF data plane component needs to contain a P4Runtime server module. In short, IML runs either on the host or in a K8S pod as a service and connects to the NF data plane components via the dedicated management networks, using P4Runtime interface.

The IML also acts as a proxy between the control plane component of each network function data plane. This allows IML to carry out seamless optimization in the data plane (e.g., running multiple instances of the same data plane component and distribute the rules and load among them, etc.). The NF control plane communicates with IML through P4Runtime protocol with optional annotations used for data plane optimization (e.g., tagging which rule belongs to which UE or is a general rule needed in each data plane instance). The NF control plane can use its custom protocol (custom NF-specific northbound API) to communicate with other components in a Service-level management network.

5.2 Cloud-Native Data Plane Mechanisms

As presented in the previous section, IML acts as a proxy between the data and control plane entities of each deployed NF instance. In this way, IML can seamlessly optimize the data plane by selecting the appropriate target, running multiple instances of the same data plane, etc. If NF control plane delegates a rule to a data plane object, IML can decide which data plane instance needs the given rule.

To control rule delegation to different data plane instances, for each NF we need to define a load balancing key, a unique identifier that groups packets belonging to the same flow, user, or application, and can be extracted from the packet headers (e.g., an IPv6 address). In case of rule injection and modification, IML will determine the data plane instance of the NF being responsible to handle the traffic belonging to the load balancing key, and load the rule to the appropriate instance. Note that there may be global rules needed by every instance. The modification of these rules affects all the NF-DP instances.

Another example, if the traffic volume of some users in their peak hours can only be served by a hardware data plane and thus their data traffic needs to be redirected to a hardware instance. In this case, IML will automatically move the states of this heavy user (called heavy hitter) to an instance running on a hardware accelerated programmable data plane. According to the load balancing key, IML will know which rules belongs to each user and can perform the migration without the direct intervention of NF control plane. Note that hardware data planes reach much higher packet processing rates without extra latency costs, but they also have limitations (SRAM/TCAM sizes, etc.).

Finally, hardware P4 programmable data planes cannot support shared usage and virtualization by default. However, the good utilization of costly hardware data plane resources requires the support of multitenancy and the share of data plane resources among different data plane programs. In DESIRE6G, we introduce methods to adapt the cloud native paradigm in its unified programmable data plane layer.

Figure 31 shows the high-level overview of the load optimizer data and control plane components that hide the NF data plane deployment and optimization details from the corresponding control plane entity of the given NF. The main goal of this component is to provide the NF control plane with a simple view about the data plane entity it controls. We assume a common control plane API through the table manipulations and the control of other data plane object can be done. In DESIRE6G, we use the P4Runtime protocol for this purpose. This logical view is implemented by the load optimizer control plane proxy (part of IML) that provides a P4Runtime-based northbound API for the NF's control plane, mimicking a single data plane instance to be managed. Note that we use an extended P4Runtime protocol where a load balancing key annotation is assigned to each rule/table entry to be inserted into data plane objects. The key is used to group rules that belong to the same traffic aggregate (e.g., a specific user, application, etc.). This extra information helps in identifying the rules to be moved if the system directs the traffic aggregate denoted by the key to another data plane instance. The load optimizer control plane persistently stores the data plane configurations given by the NF control plane and forwards the configurations to the appropriate NF data plane instances via standard P4Runtime protocol over gRPC It also manages the load optimizer data plane component that is responsible for heavy hitter (or generally hardware) offloading and load balancing among the different NF data plane instances.

The load optimizer data plane implements two key mechanisms; **heavy hitter offloading** and **load balancing**. The heavy hitter offloading logic is applied first to determine if the packet needs to be forwarded to a hardware-based NF data plane or can be served by a software-based implementation (e.g., running as a CNF). Offloading logic also handles changes in the heavy hitter (HH) state during operation. For example, if a user becomes a HH thus its throughput demand increases significantly or it runs a low latency application, the load optimizer control plane is notified, and the state migration process starts to prepare the hardware-based NF data plane instance for receiving the user's traffic. Similar actions are needed if the user becomes a non-HH. In addition to hardware offloading the load optimizer data plane also handles load balancing among different NF data plane instances. For example, software data planes run in containers (as CNFs) have different limitations related to throughput and latency. However, in K8S we can easily scale up and down the number of instances, so some users may have to be migrated to other instances in run-time. The load optimizer handles traffic steering in case of up- and down-scaling and the load optimizer control plane proxy solves the data plane state migration seamlessly.

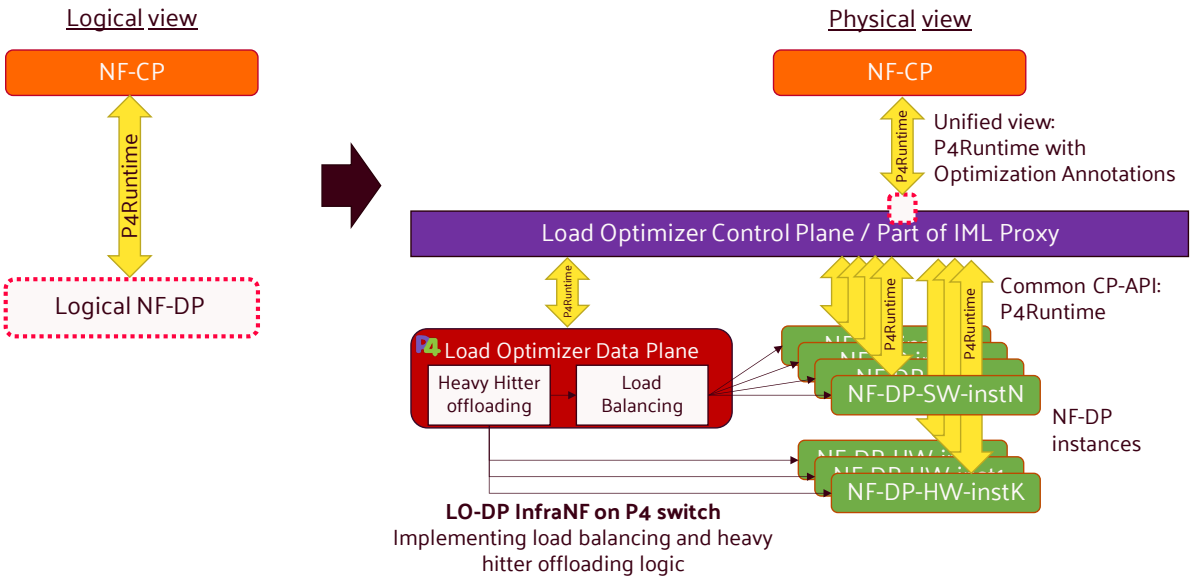

FIGURE 31: LOAD OPTIMIZER COMPONENT FOR SEAMLESS LOAD BALANCING AND HARDWARE OFFLOADING

Existing P4 programmable hardware targets are only capable of executing a single P4 program per pipe. Multi-tenancy in its traditional means is not supported natively by these devices. In DESIRE6G, we handle **multi-tenancy on P4-programmable targets** by aggregating multiple P4 programs and adding classification and resource management logic to the aggregated P4-program. To provide isolated access to data plane objects and hide the collocated data plane programs, a control plane proxy (part of IML's proxy functions) is also introduced. This component provides the NF control plane with a simple logical view of the NF data plane via P4Runtime API. The proxy also uses P4Runtime to configure the aggregated P4 pipeline running in the hardware target, and maintains the rules added to the data plane objects in a persistent storage (Redis Database in the current implementation). In case of data plane reinitialization (e.g., if a new P4 program is added to the aggregated program), the state before the update is restored automatically, being hidden from the NF control plane components. The program aggregation-based multitenancy support is highlighted in Figure 32.

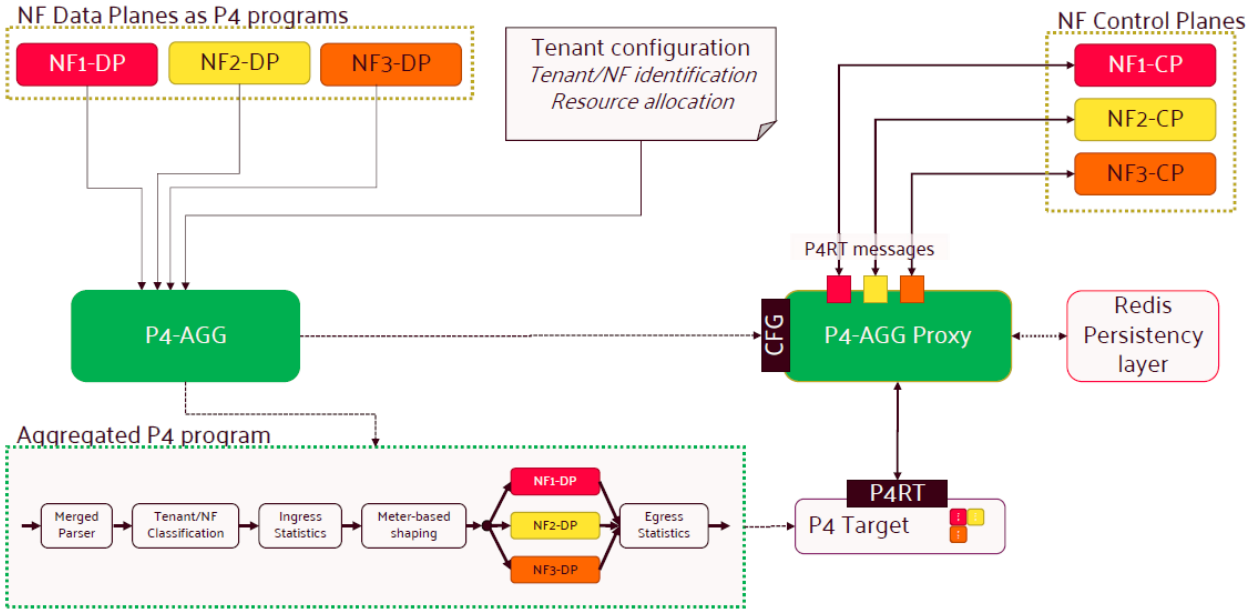

FIGURE 32: MULTITENANCY SUPPORT WITH P4 PROGRAM AGGREGATION (P4-AGG)

The main ideas were also described in patent applications [48], [49], [50].

5.3 Network Service Deployment and Operation

Service deployment is handled by the SMO. As mentioned previously, it disaggregates the service graph to subcomponents and assigns each subcomponent to a DESIRE6G site. In each site the IML implements local network function virtualization orchestrator (local NFVO) functionality. This local NFVO is responsible for handling the available resources at the given DESIRE6G site, perform local optimization through selecting the proper resources to implement the network service subgraph, deploy both network and application functions and configure networking including the virtual management networks and the data network. The configuration of the data network includes the setup of various infrastructure network functions. The structure of the local NFVO is depicted in Figure 33. The local network service descriptor (localNSD) is created by SMO and handed to the local NFVO component of IML to initiate the deployment of the subgraph on the specific site. The localNSD relies on an existing VIM (e.g., K8S in our case), plugins for implementing the deployment process on hardware data planes and network configuration tools. The localNSD lists the NF and AF instances to be deployed. Each NF instance implements a network function defined as a network function description (NFD). In Desire6G the NFD defines the control and data plane part of the NF, the main ports of the data plane component, the protocol to its control plane and the high-level interfaces of the NF control plane. In contrast to prior

approaches, the data plane part in the NSD does not carry any implementation-specific details. Data plane implementations are described in network function implementation descriptions (NFIDs). The implementation descriptors carry information on the target specific binaries and their configuration and deployment process. An example for the local NSD required by IML is shown in Section 11 (Annex I).

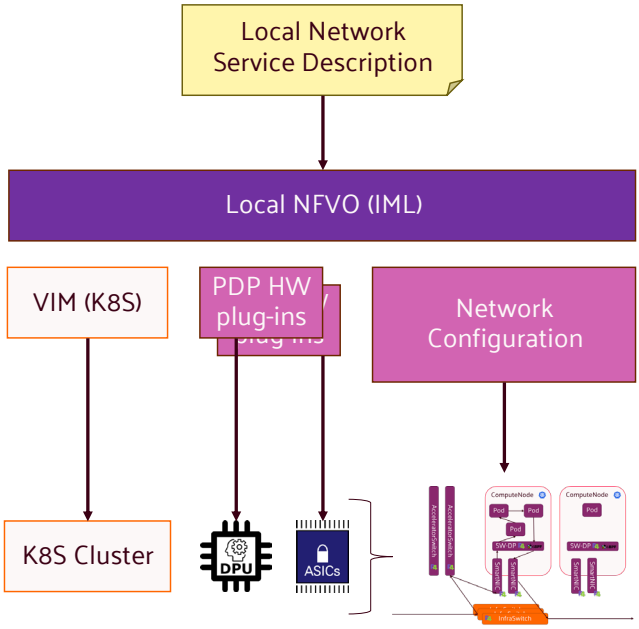

FIGURE 33: LOCAL NSD DESCRIBES A SUBGRAPH OF THE SERVICE GRAPH

5.4 Infrastructure Functions

As the packet processing infrastructure is programmable in DESIRE6G, we can encode extra information into packet headers that can be taken into account during packet processing and forwarding. We introduced a DESIRE6G header structure to store the necessary information needed for routing along the network service graph or handling QoS. The main DESIRE6G header structure is depicted in Figure 34. The DESIRE6G header can encapsulate IP packets, directly inserted after the Ethernet header. The main header is 9 bytes and contains the following fields: 2 bytes of serviceId which is a unique identifier of the network service instance (i.e., network slice), 2 bytes of locationId is the RAN site location of the UE in case of an edge-to-edge service. 1 bit of hhFlag used to express the activity of the user the packet belongs to. It is set if the UE is a heavy hitter and unset if not. This field is followed by 7 reserved bits for future usage. 2 bytes of nextNF describes the next network function to be executed on the packet, while 2 bytes of the nextHeader field points to the next protocol data unit located after the DESIRE6G header.

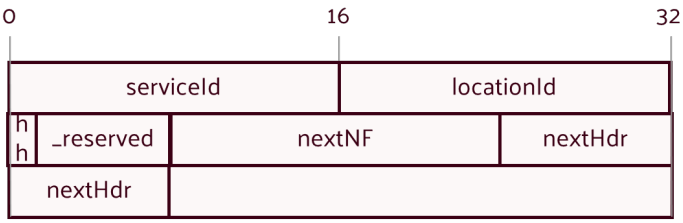

FIGURE 34: THE MAIN DESIRE6G HEADER

The DESIRE6G header is used by the following infrastructure network functions (infraNF).

UE2Service Mapper: The DESIRE6G header is attached to the packet by the UE2Service Mapper infraNF. This component receives packets and maps the IP address or other identifiers to services. In DESIRE6G, we assume that IPv6 addresses can represent the network service specific identifier of the UE. Note that a UE can be attached to multiple network services at the same time, resulting in multiple IP addresses to the UE. UE2Service Mapper infraNF will add the DESIRE6G header to the packet. It fills the serviceId, locationId according to the network service to be applied and the RAN site location, the nextNF is set to the first non-infraNF to be applied, according to the service graph. The hhFlag can also be filled if heavy hitter offloading is supported by the network service, and heavy hitter detection is activated in the service graph. The control plane of the UE2Service Mapper provides a REST-based northbound API (see Table 2) through which the network service can be created and configured.

TABLE 2: UE2SERVICE MAPPER API CALLS

/api/service/create	POST	direction (uplink/downlink), UEIDs (IPv6 prefix), serviceId, initial nextNF (first NF in the forwarding graph)	Creates the service and fills the data plane objects of UE2Service Mapper.
/api/service/delete	POST	serviceId	Deletes the service.
/api/UE/setHH	POST	UEID (IPv6 address of a heavy hitter), state (True/False)	Sets the heavy hitter flag for a specific UEID
/api/UE/attach	POST	UEID (IPv6 address), locationId (RAN site)	Activate a UEID and attaches it to a RAN site

/api/UE/detach	POST	UEID (IPv6 address)	Deactivate the UEID and removes from the RAN site
/api/UE/move	POST	UEID (IPv6 address), locationId (RAN site)	Moves a UEID from one RAN site to another, i.e., handover.

NF Router: NFRouter implements the forwarding logic needed for traversing the forwarding graph of the network service. It ensures the delivery of packets from one NF to another. The NF is identified according to the nextNF field in the graph. This data plane component forwards the packet to the NF-DP which processes the packet and then send it back to the NF Router. Then the NF Router updates the nextNF field according to network service graph and the exit state of the previous NF (the output port in the abstraction mentioned above). Finally, it forwards the packet to the nextNF. This component is also responsible to remove the DESIRE6G header before forwarding the packet to application functions or network domains not supporting this header type. See Table 3 for the API calls of the NF Router.

TABLE 3: NFROUTER API CALLS

/api/add-nf-forwarding	POST	serviceId, locationId, NFId, port, vlanId, decap-mode	Sets forwarding to the instance of nextNF in the service graph. If the nextNF points to an AF, it removes the DESIRE6G header.
/api/del-nf-forwarding	POST	serviceId, NFId, locationId	Deletes the forwarding rule.
/api/add-next-nf-rule	POST	serviceId, locationId, NFId, port, vlanId, nextNFId	Adds nextNF rules to configure the links between two consecutive NFs. After the execution of an NF

			is finished, the nextNF is set.
/api/del-next-nf-rule	POST	serviceId, locationId, NFId, port, vlanId	Deletes the nextNF rules.

Transport Adapter: The infrastructure may be composed of multiple DESIRE6G sites with different transport technologies between them. The Transport Adapter infraNF is responsible for tunnelling between two DESIRE6G sites using one of the possible encapsulation protocols. The logic of transport adapter is protocol independent and can rely on various protocols like GTP, VXLAN, etc.

6. Main functions and message flows

Message flows describe the system's behavior during different functional steps. In this document we defined 3 message flows that are used to show the introduction of a new service in 3 steps: i) the definition and translation of the service, ii) the deployment of a service and iii) the attachment process of a user to a given service. We also described a 4th message flow to show an example of how a traditional 3GPP procedure (mobility) can be handled in the DESIRE6G system, and we also introduce some steps without message flows to give a comprehensive overview of the system processes.

6.1 Bootstrapping

Pre-deployment or bootstrapping has multiple stages. On an "empty" DESIRE6G system, first the SMO needs to be started – this can use normal data center methods, e.g., applying a deployment file in Kubernetes to deploy the mandatory components of the SMO (Figure 35).

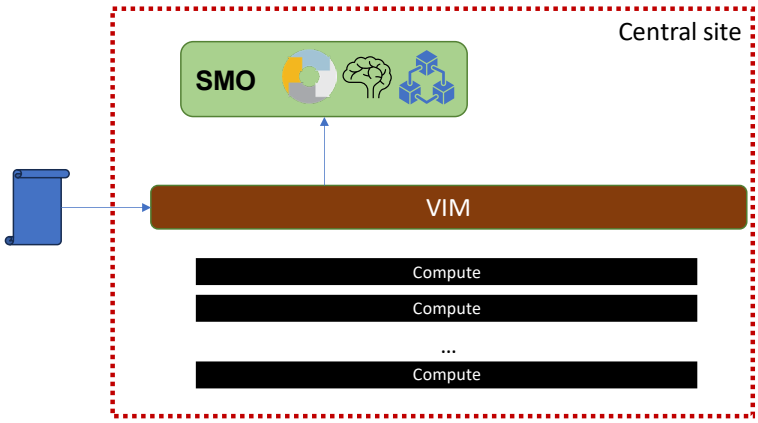

FIGURE 35: FIRST STEP FOR STARTING A NEW DESIRE6G SYSTEM

Once the SMO is running there are several next steps that have to be repeated for each site where DESIRE6G components are planned to be running (see Figure 36):

1. Start the VIM that controls the compute resources.
2. Start the IML service for the given site – this could also be started as a normal k8s service.
3. Configure the IML with the site-local resources. There are multiple options:
    a. (Semi-)Static configuration with the available resources and their access, e.g., address of the local physical P4 switches and VIM. The compute resources are handled dynamically by the VIM.

b. Registration-based scheme: the local P4 resources register to the IML after starting up (either via some broadcast mechanism or (pre-)configured IML address). Note that only the resources not visible for the VIM should implement such methods.

4. Connect the IML to the SMO: the IML must send a registration message to the SMO service.

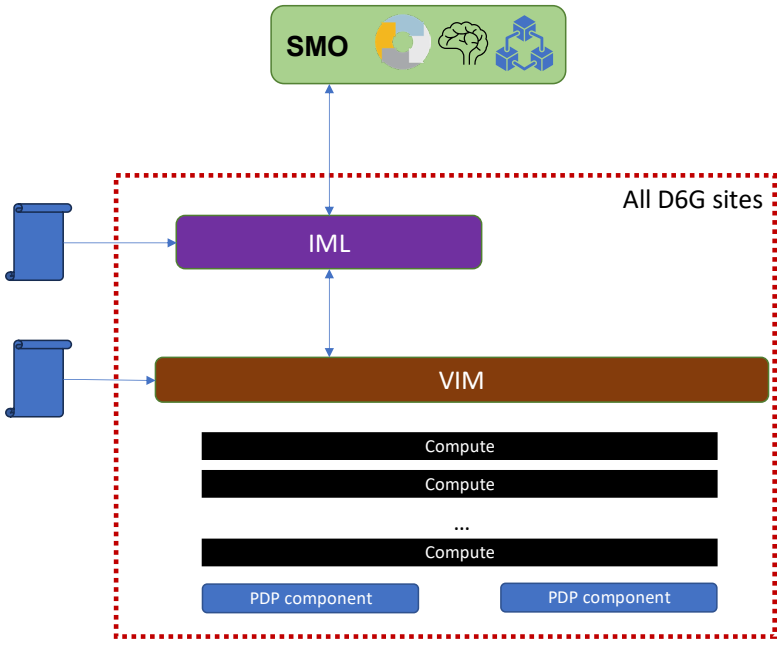

FIGURE 36: ADDING A NEW SITE TO THE DESIRE6G SYSTEM

After adding the new site, the site router needs to be configured to allow site connectivity. There might be multiple connections between sites. The IML will set the "next site address" only, routing between the sites is handled by either the SDN controller or some other transport mechanisms (e.g., routing protocols). MAS can alter the site-to-site links dynamically to optimize e.g., end-to-end latency.

6.2 Service creation

Services in D6G are defined by their service graph and additional configuration parameters, such as QoS requirements, preferences (e.g., prefer low latency over all other Key Performance Indicators - KPIs) and policies (e.g., geographical or time-of-day limitations). A service graph is an abstracted view of a service that specifies the set of required NFs as well as the order in which they must be executed. The first step in creating a service is the creation of the service description from various external sources, e.g., SLAs. This is depicted in Figure 37.

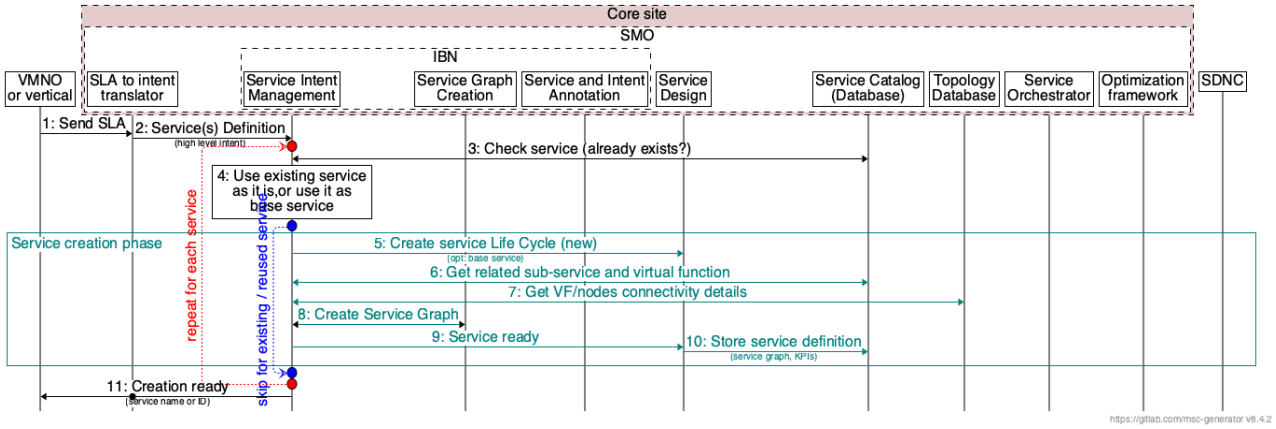

FIGURE 37: THE SERVICE CREATION PROCESS

The process starts with an external request, which typically comes in the form of an SLA, but other formats are also possible as long as the translator logic is able to understand it and create the intents. After translating the input into the internally used intent structure, the service management and policy framework - the heart of the service creation process - starts working on the input. First it checks whether the required service already exists, then (if not) create the service description. The service description contains the service graph, required KPIs and potentially other policies.

Upon successful creation the service description is saved into the service catalog. The process might end here, but if the SMO has further input e.g., on QoS requirements towards transport, some further steps can be made. After these steps, the SMO can send an acknowledgement to the party who requested the creation of the service. After this point a service deployment request for the created service can be sent and will be executed by the SMO.

The exact steps are the following:

1. Receiving a high-level description (e.g., in the form of an SLA) of the service: what is intended, KPIs, etc.
2. The SLA is translated to intents that are processable by the Intent Management Framework.
3. The existence of the service is checked, using the Service Catalog, which stores all available services in the system.
4. If we have an existing service suitable for the given input, there are options to fully or partially re-use that service. In case of direct re-use, skip steps until step 12.

5. The system enters the service creation phase, after which the Service Catalog will be populated by the service description. The Service Design function gets a trigger that a “new” service is to be created. If we use an existing service as a base service for the new one, that is also reported.
6. If there is a base service, this step is needed to get the service description of that service.
7. If there is a base service, this step is needed to get the topology (site-level) of the base service.
8. With all the necessary information available the Service Graph Creation entity can generate the service graph for the required service.
9. The Service Design block gets the service descriptor with a “service ready” message.
10. Upon that it stores the service definition (service graphs + associated KPIs) in the Service Catalog.
11. Then replies to the original request with a “service creation ready” message that contains a pointer (name or ID) to the corresponding DESIRE6G service.

After these steps the service is ready to be deployed in the system. The created service description can be used right now or at any later time when the service is needed.

6.3 Service deployment

Service deployment is described in Figure 38 in detail. During deployment the first step is to select which sites are needed for the service. This is done by the SMO and its optimizer algorithms using all types of constraints, policies and - if available - end-to-end KPI descriptions like latency or bandwidth. Some network functions will have their own constraints, e.g., the RAN lower layers need to be close to the actual UEs. Steps 1-4 describe some basic checks, while steps 5-11 describe the decomposition and site selection process as detailed below:

1. The Service Orchestrator gets a “deploy service” request on its external API. Note that the caller might specify the access sites where the service is required, but this is not mandatory, as the access sites can be dynamically set too based on actual connectivity requests. If no access site is presented at this step the system only deploys the core / central components and the access sites will be selected and populated dynamically. On the API it is also possible to specify an “all site” flag in which case the access part will be populated to each and every access site in the DESIRE6G system.
2. It checks whether the requested service (name, id) is present in the Service Catalog.

3. The Service Catalog replies with the corresponding service description, or it sends an error message if the service is not present.
4. In case of error, the SO reports it to the caller.

If the service exists in the system, the decomposition phase starts:

5. The SO informs the Intent-Based Service Management to start decomposition.
6. Which in turn downloads the (site-level) topology of the DESIRE6G system. Note: this step might not be needed as the topology can be cached or the IBM can use a pub/sub method towards the Topology Database.
7. After having all necessary information, the IBM will instruct the Optimizer to do the actual decomposition.
8. The Optimizer will run its optimization algorithms to find the proper site(s). It can use various inputs for this decision e.g., from the service description (connectivity, KPIs), from the SMO or MAS (e.g., processing or network load information), etc.
9. Upon having a good result that is communicated with the IBM in the form of an annotated service graph.
10. The IBM will generate sub-intents for the different domains that are needed for the service (e.g., radio, transport) – the sub-intents are sent to the respective domains in this step, this is not showed in detail on the message chart.
11. After this step the annotated service graph is sent to the SO which can start the physical deployment

After step 11 the SO is ready to deploy the service to all sites where it is needed. It has enough information on the selected sites and the required sub-service graphs per site with their (sub-)KPIs.

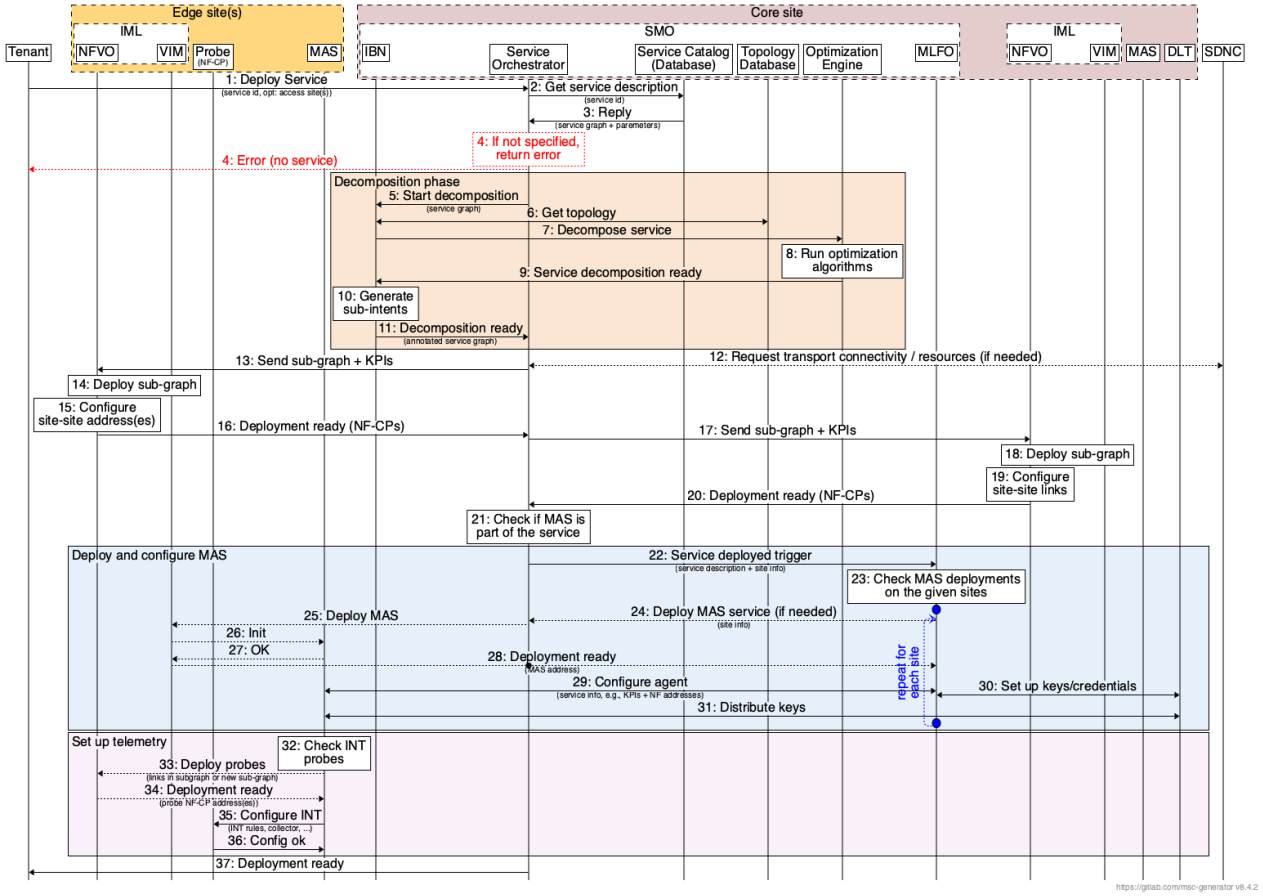

FIGURE 38: THE SERVICE DEPLOYMENT PROCESS

After finding the proper sites the SO instructs the corresponding IMLs on each site to deploy the sub-graphs accordingly. It might also instruct the SDN controller to create transport connectivity and reserve resources – note: reservation is rarely needed as transport is usually not the bottleneck, but we keep this step anyway here as a future possibility.

In the physical deployment phase (steps 13-20) the logical view of a service with its service graph is transformed to an intermediate view with multiple infrastructure functions such as load balancers or transport adapters. These are needed for the actual physical deployment. An example can be seen in Figure 39.

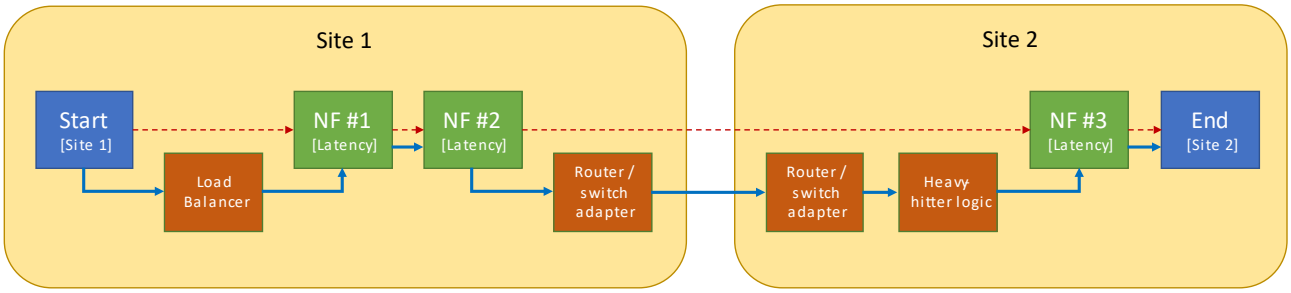

FIGURE 39: INTERMEDIATE LOGICAL VIEW OF THE SYSTEM DURING DEPLOYMENT

After this intermediate logical view is available, the IML can do the actual physical deployment at each site (see Figure 40 for illustration). The IML will put together the combined NF-DP code wherever it is possible and ask the VIM to start workers.

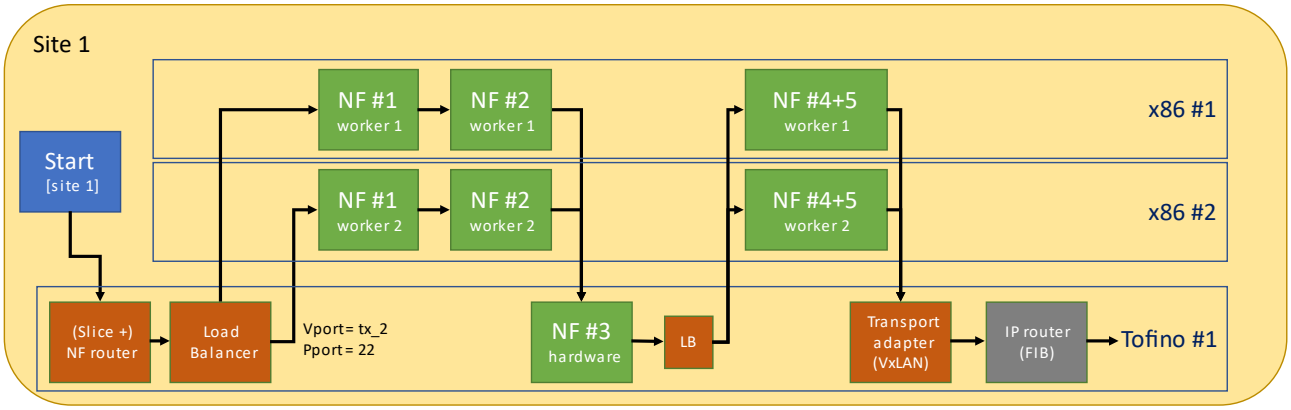

FIGURE 40: LAST DEPLOYMENT STEP WITH THE PHYSICAL VIEW OF THE SYSTEM

After this phase it is possible to instantiate the service-specific MAS (blue box in Figure 38) and required telemetry functions (purple box in Figure 38), as described below:

21. The SO checks whether MAS is part of the service or not. There might be some legacy service that will not require MAS, this is the reason keeping it optional.
22. If MAS is needed, upon physical deployment ready the MLFO gets a “service deployed trigger” which contains the annotated service description together with the selected sites.
23. The MLFO checks all sites where any components were deployed and on each site it checks if there is a running MAS instance
24. If not, it initiates deployment of MAS on the given site (usually this is not needed). The deployment is done via the local IML/VIM.

Note that MAS will be a service, meaning it will have one address on a given site (e.g., VIP, service address), which can be used for external entities to communicate with it.

29. After MAS is up and running the MLFO configures it by sharing necessary live service information with the MAS agents: KPIs, running network functions (CP), etc.
30. The MLFO asks the DLT to set up secure connectivity between the MAS entities.
31. The DLT shares the keys for the MAS agents.

Telemetry extensions can be part of the service template, but the MAS can also request telemetry functions during the attachment of users or on the fly. This way it will be possible to execute user-specific end-to-end telemetry collection used e.g., for root cause analysis. If telemetry is part of the service template the necessary functionality is requested by the MAS agents. The details are the following:

32. MAS check whether the required probes are deployed or not.
33. If not, it instructs the IML to deploy them. For this the IML needs information on where the given probes should be deployed. This information can be conveyed in multiple ways: either by specifying links in the sub-graph or by enriching the sub-graph and send that to the IML.
34. Once the probes are deployed the IML informs the MAS.
35. The MAS then configures the INT probes by specifying rules and optionally a telemetry collector. Note: here the MAS talks to the respective NF-CP entities directly.
36. Upon successful configuration the probes report it to the MAS. The system can decide that the entire service need to wait until this step or the “deployment ready” message can be sent together with the MAS configuration step (as the data plane is already up and running). The default behaviour is to allow the service to start once the necessary NFs are deployed. Strictly controlled services will wait until the MAS configuration step is also finished.

Note that if the “dynamic” approach is used, meaning that the necessary NFs and AFs are deployed on a per-request basis (i.e., only upon request from a UE), then whenever a new UE attaches to the service the UE controller (e.g., Session Management Function) needs to make sure that the proper service functions are already up and running on the requested sites. If not, it informs the SMO to do a (partial) service deployment. Before this step (UE attach) the endpoint locations are unknown, so end-to-end latency related configuration (e.g., site selection, routing, latency budget distribution) is not possible.

6.4 Service attachment

A service can be selected by the UEs at any time after a given UE is connected to the network. A UE may be active in multiple services at the same time. The simplest mapping inside the UE is via using different OS level interfaces, but other types of mapping are also possible, especially if the UE can run its own D6G stack. From the QoS point of view, a service is the highest layer in the QoS hierarchy as described above. It is possible to set different QoS classes inside one service slice, but the service itself will also have parameters that will be used to describe QoS treatment (like end-to-end latency) meaning, that if the different flows of an application would require very different treatment it is better to map them to different services.

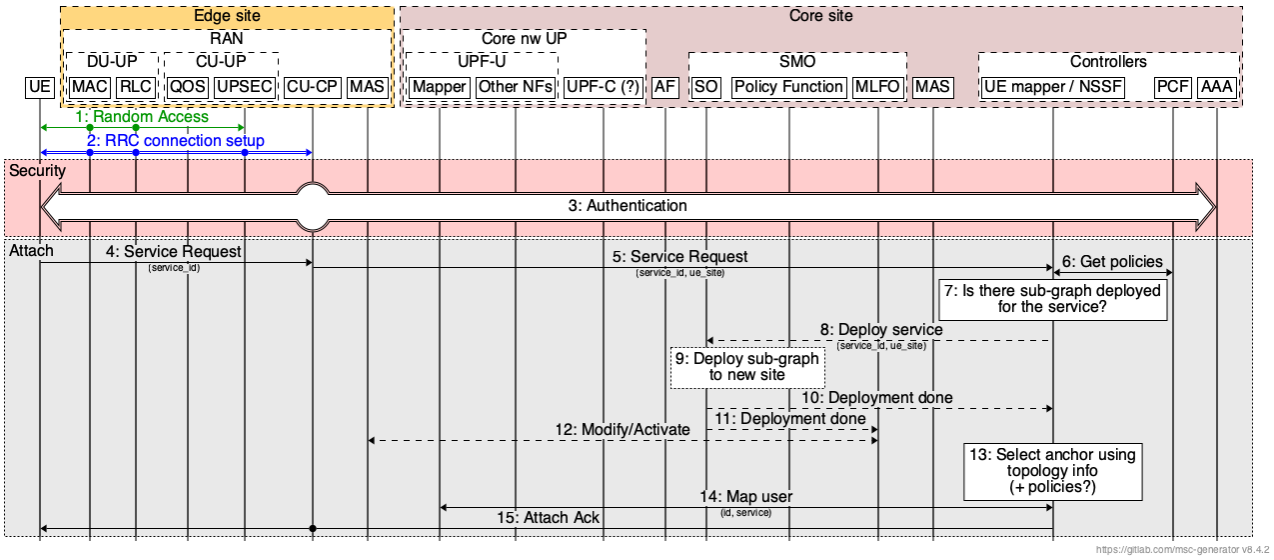

FIGURE 41: USER ATTACHES TO A SELECTED SERVICE

The UE starts its lifecycle with a generic connect procedure (steps 1-3), after which it has a basic security association with the network. Using this the UE sends a service request (4) to the network with the proper service ID. The RAN will relay this message to the core network (5), specifically to the UE mapper (or NSSF) service. After this step the network walks through the followings:

6. The UE mapper can download policies for the given UE
7. It checks whether the service (more specifically its edge components) is already deployed on the given site where the UE is active
8. If not, it asks the SO to deploy the service on the given site
9. The SO will deploy the required service sub-graph to the edge site
10. Then sends a ”deployment done” message to the UE mapper

11. And informs the MLFO about the change
12. The MLFO might need to modify or activate the MAS agent on the new site
13. After / if the sub-graph is deployed the UE mapper selects the proper anchor (NF-CP)
14. And maps the new UE in the anchor's mapper NF (both the service ID and the UE location)
15. After these steps are done it signals the successful service activation to the UE

6.5 Service Assurance - MAS

This section focuses on the message flows describing MAS operation during service lifetime, with the aim to detect/anticipate service degradations and perform service reconfiguration. In contrast to service creation, deployment, and attachment workflows, MAS-based service optimization is specific of the service type and use case under evaluation. For illustrative purposes, we concentrate the QoS service assurance reconfiguration based on intelligent routing use case presented in Section 4.2.1. Figure 42 shows a generalization of a the service reconfiguration triggered and performed by MAS in the example of Figure 25. In particular, it focuses on service flow re-routing performed at the site A with the telemetry data collected and processed at both local site A and remote site B, in order to keep the desired e2e QoS within the network segment.

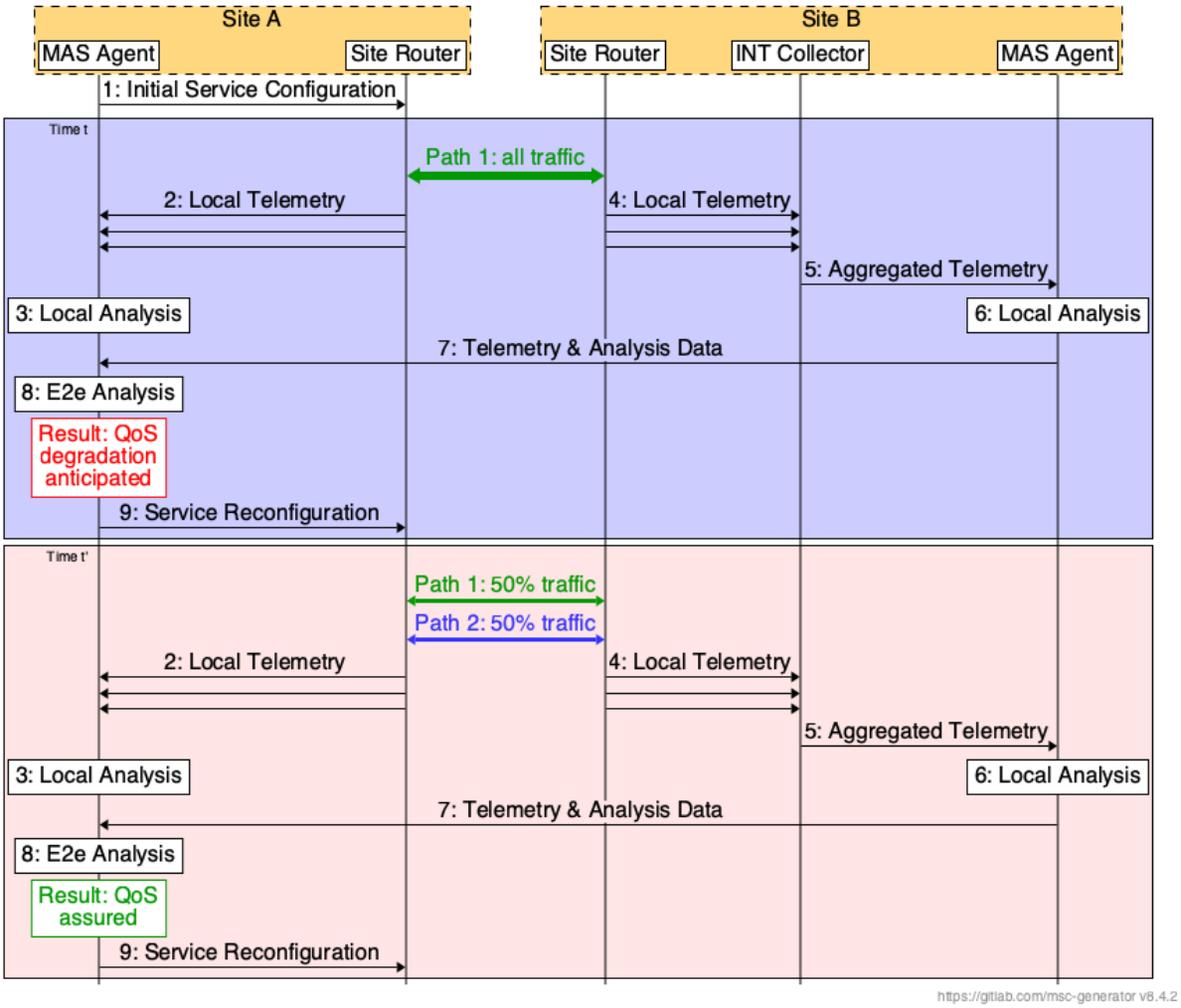

FIGURE 42: SERVICE RECONFIGURATION PERFORMED BY MAS

Assuming that proper MAS agents are deployed and active in each of the sites, the workflow is initiated as follows:

1. An initial configuration of the service is setup. Following the example in Figure 25 and assuming that a delay budget for the network segment has been configured in the MAS agent of Site A, this agent configures an initial routing policy in the site router. In the embedded topology of the example at time *t*, it can be seen that all service traffic will be routed through one of the two pre-computed routes.

Then, the following steps are continuously repeated with a periodicity defined during MAS configuration (e.g., every second). The steps are showing the operation at time *t*:

2. At site A, local telemetry is collected at MAS agent from site router, which is done according to the telemetry configuration (see steps 32-36 of service deployment workflow in Figure 38)
3. The data collected at local site A can be used at this stage to perform some local analysis, e.g. aggregating and characterizing collected telemetry (traffic volume) to facilitate its posterior use for e2e analysis.
4. At site B, a similar operation of local telemetry collection is performed. To distinguish from previous local telemetry collection at Site A, this process allows collecting delay measurements that will be used for e2e analysis and decision making.
5. In line with the service monitoring schemes and cases of study in section 4.2 and due to the fine granularity of delay measurements performed by INT probes, an INT collector receives raw telemetry and generates aggregated and processed telemetry values. The result of this aggregation is passed to the MAS agent in charge of controlling the service at Site B.
6. Similarly to step 3, local analysis can be performed.
7. With a given configured frequency, the MAS agent at site B sends telemetry and analysis data to the MAS agent at site A. Among relevant metrics, those related with QoS assurance such as delay (in terms of aggregated measurements such as average and maximum) are included as part of the shared message.
8. Upon the reception of telemetry and analysis data from Site B and with the addition of the local telemetry data collected at Site A, e2e analysis is performed in order to evaluate whether the committed QoS of the service is assured or conversely, it is likely to happen a QoS service degradation, i.e., measured delay is close or exceeds the required budget. The latter is the case illustrated in this step, which triggers service reconfiguration.
9. The actions decided by the MAS agent to mitigate the service assurance violation are sent to the element that must be reconfigured. In particular, for this example use case, a new routing policy for the service flow is sent to the site router. After applying the new routing policy, service traffic is now routed evenly through both pre-configured routes.

The rest of the workflow shows the same operation for next time period *t'*. As can be observed, after e2e analysis, the MAS agent validates the received performance as tolerable and, as a consequence of this, no re-routing actions are needed to guarantee the committed QoS.

6.6 Service Assurance - SMO

If the MAS framework cannot solve the KPI violation issue with its own resources, it will signal the problem to the highest layer to the SMO. The SMO can then decide on a mitigation - if there are any. The message flow of this event is quite similar to the service deployment case with a few simplifications:

- Since the service is already running, the SMO does not need to deploy or configure the MAS
- In most cases the problem only affects a given site or a few sites, in which case a partial redeployment is enough

However, in some cases, especially if the resources are already highly utilized, this process can affect multiple services, in which case all the affected services will have to go thru a full or partial redeployment as depicted in Figure 43.

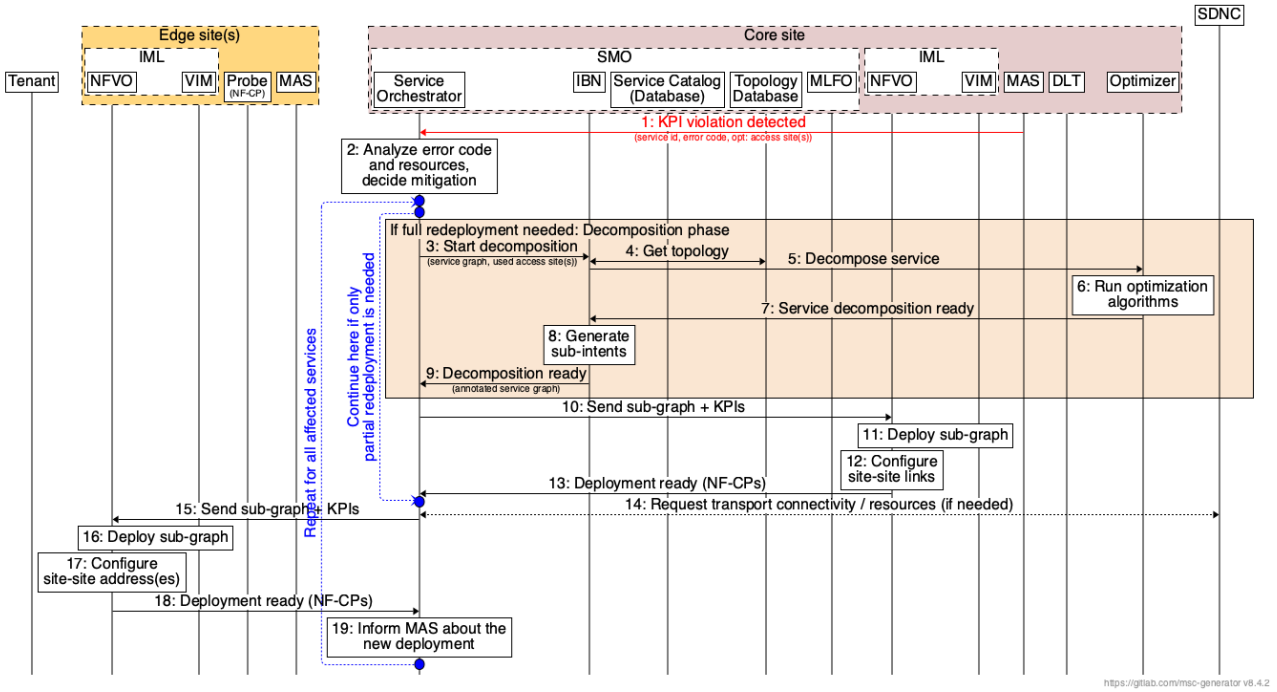

FIGURE 43: PARTIAL OR FULL REDEPLOYMENT OF A SERVICE OR SERVICES

Step 1 is new compared to a normal deployment. This is the trigger for the redeployment process, arriving from a MAS agent (most likely the agent running at the central site). It is an error message telling the SMO that there is a significant problem that the MAS framework cannot solve. The message contains a service ID and an error code (e.g., ”insufficient resources”, ”site down”), plus optionally access site identifiers - which suggests that the MAS already pinpointed the location of the problem.

The rest of the process follows the deployment steps (see section 6.3 for details) with some differences as discussed above.

6.7 Mobility

With the proposed architecture any mobility event can be handled automatically using either learning or some in-path update messages generated by the target (new) RAN site upon having successfully obtained the UE from the source (old) RAN site. Figure 44 shows the learning-based option.

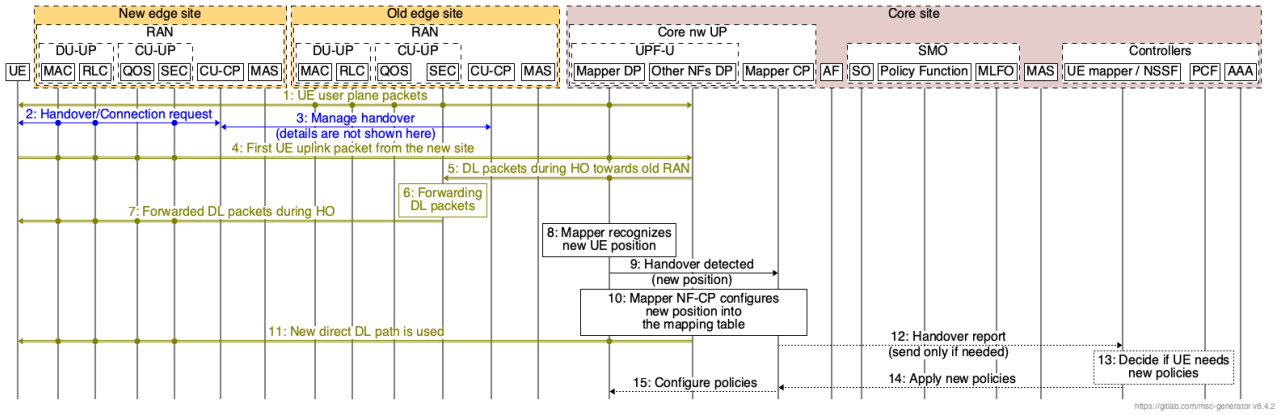

FIGURE 44: HANDOVER HANDLED AUTOMATICALLY

The UE packets are flowing through the old edge site in the beginning before handover is started. Upon handover the RAN entities are working as they used to be, either there is UE-initiated or network-initiated handover, steps 2-3 are the same as the current state of the art.

After the new RAN node has the UE, it starts sending uplink packets to the UE mapper in the core site (step 4). The downlink in this case is still flowing thru the old RAN node (5), where forwarding (6) is usually set towards the new RAN node (7) – this can be subject to service configuration, simple, cots-optimized services might not need this.

Step 8 describes the automatic process: the UE mapper can run a verification for each packet whether it is coming from the location that is stored in the internal mapping table. If not, the mapper DP can generate a notification towards its CP counterpart, which in turn modifies the mapping table with the new location. After this, the DL path also uses the direct route thru the new RAN site.

After the user plane is set correctly, the UE mapper CP can inform the 3GPP controllers about the handover. This can be optional, may be driven by policies, e.g., “only inform the controllers if the UE

changed tracking area". After the notification the 3GPP controllers can decide if there is a need for new user plane policies, e.g., different charging profile.

Note that additional inputs for mobility (like monitoring information, see in section 4.2.3) may be utilized for deciding when mobility should happen, plus additional outputs from the mobility event could be utilized by other components, e.g., to further optimize the end-to-end graph for the given UE.

7. Mobile network functions

7.1 Radio Access Network

The starting point for the DESIRE6G RAN is 5G Standalone 3GPP with the O-RAN Alliance extensions of the RICand SMO network functions, the additional O-RAN Alliance APIs (like O1, O2, A1, E2) and the Lower Layer Fronthaul split between DU and RU, This is illustrated in an SNPN (Standalone Non-Public Network) context below in Figure 45.

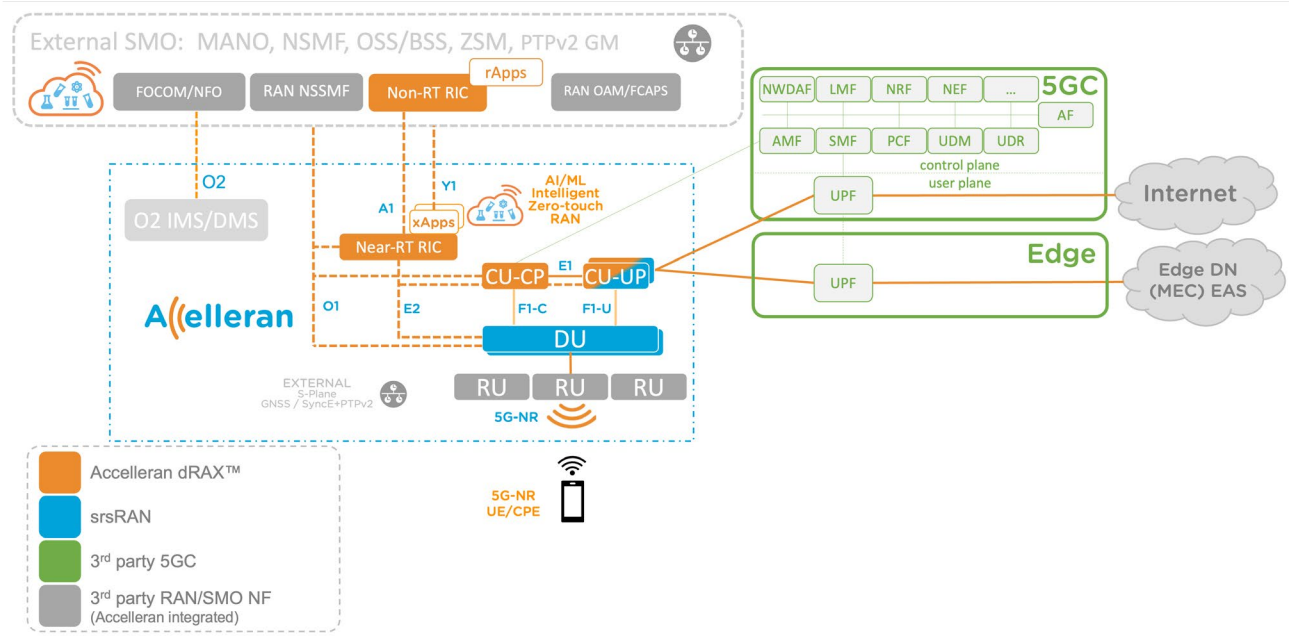

FIGURE 45: D6G OPEN RAN BASED SNPN CONTEXT

This Figure 45 is derived from the Accelleran Open 5G SNPN solution, combining the Accelleran dRAX™ network functions (Near-RT RIC with xApp SDK, CU-CP, CU-UP. This solution is the starting point for DESIRE6G WP4, where the srsRAN network functions provide a complete Open RAN aligned gNB, and some such as the CU-UP will be extended to allow HW acceleration research of PDPC/SDAP, especially with enablement of the security option, shown below in Figure 46. Together with an open-source 5GC and the dRAX™ SMO dashboard, containing the elements of the O-RAN SMO needed to configure and operate the Open RAN, an operation open 5G SNPN will be configured (such as on the ARNO testbed).

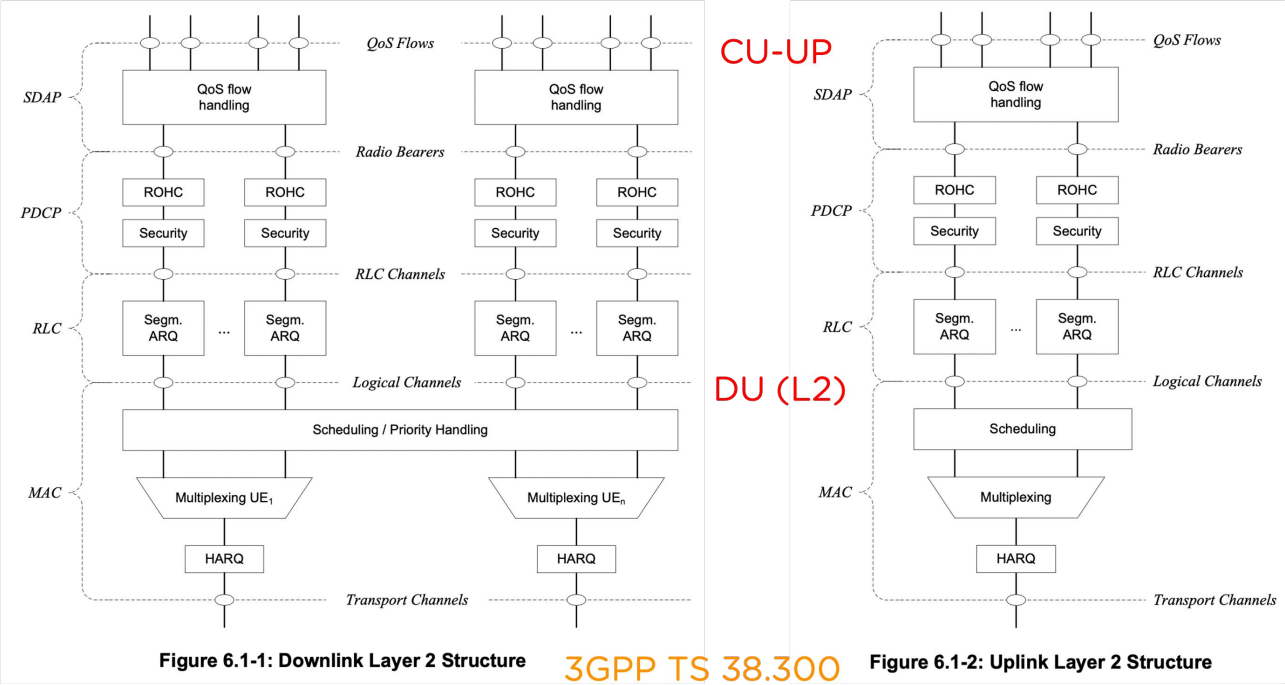

FIGURE 46: D6G OPEN RAN USER-PLANE CU-UP/DU NF BREAKDOWN [20]

As described in 3.1 and illustrated in Figure 9, there are many expected evolutions of the 5GS towards 6G, including in the RAN. The architecture and design of this evolved 6G RAN and telemetry sub-system, will need to support the following "southbound" technologies (this section discusses the "soutbound" evolutions, with the "northbound" evolutions discussed in 3.1.1), repeated in Figure 47, in addition to the purely 3GPP 5G RAN network functions shown above in Figure 46:

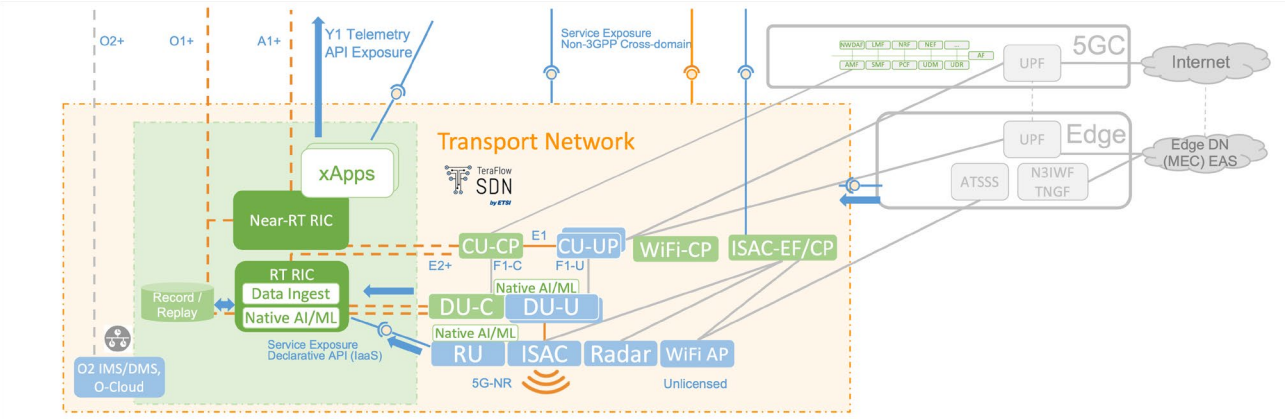

FIGURE 47: 'SOUTHBOUND' RAN EVOLUTIONS OF OPEN RAN (CONTROL-PLANE) TOWARDS 6G

A list of some of these 'southbound' RAN evolutions are as follows:

- **Advanced Non-3GPP:** Heterogeneous multi-radio access technology like Wi-Fi 7 or 8, but also ultra-wide band and LIDAR technologies. While in 3GPP 5G the focus was on UP integration towards the UPF, it is expected that the positioning capabilities will increasingly be injected into the RIC to provide enhanced cross-domain functionality, beyond SotA 3GPP.
- **Native AI/ML in DU/RU:** While the Near-RT RIC can ingest telemetry and control some aspects of the DU/RU through the O-RAN E2 service modules, the real intelligence will become a reality if it is deeply embedded into the 6G native AI/ML. This is the scope of the identified but unspecified real-time RIC (RT RIC) and their dApps, which is outside DESIRE6G scope but will be needed for future 6G capabilities.
- **ISAC:** The addition of radar-like sensing, integrated into the communication radio units (ISAC or Integrated Sensing and Communications) will require processing and usage of this sensing information, to merge with other higher-level telemetry for advanced functionality.
- **Full-duplex and integrated access and backhaul (IAB):** These technologies will extend 6G RAN coverage and will add additional interference management and bandwidth scheduling that will result in increased telemetry to be ingested and utilized.
- **Dynamic spectrum re-farming:** Will require full sensing of the channel-state information (CSI), and neighbour interference, to ingest and process the interference extracting valuable telemetry for optimization and co-existence instead of being discarded as pure noise.
- **Cell-free mMIMO:** This technology requires close coordination of DU and RU, with dynamic reconfiguration of the user-centric Access Point (AP) associations, for coherent multipoint transmit/receive and synchronization. Early research results show the need for RIC-like intelligence [15].

From an O-RAN perspective, many of these newer 6G RAN capabilities require deeply embedded control loops, with faster real-time response, leading to the need for the RT RI' (or Real-Time RIC) and their dApps.

In DESIRE6G, beyond the 6G RAN functional evolutions, above, and the service management and optimization impact in the RAN, described in section 2.4, the primary RAN research and innovations are in the WP4 programmable and accelerated data plane and supporting infrastructure layers, reported in the initial Deliverable 4.1 [42].

Starting from the 5G CUPS (Control & User Plane Separation) in the RAN, and with the industry-wide evolution from dedicated and proprietary RAN BBU (Baseband Unit) appliances towards software-defined virtualized and later cloud-native RAN network functions (VNF/CNF), illustrated in Figure 47, running on COTS compute and network infrastructure, DESIRE6G researches a number of RAN areas focusing on the programmability and hardware acceleration of these RAN network functions and layers, in a number of significant areas:

- Service definition as network function graph, supporting dynamic programmability (see 2.3)
- Programmability of the networking hardware, such as with P4 (see 2.4.2), though current DESIRE6G research has determined that while P4 programmability is suitable for the mobile core UPF network functions, parts of the RAN stack is not feasible to be implemented with the same methods mainly due to the quite special functionality (e.g., signal processing).
- The cloud-native data plane and mechanisms (see 5 and 5.2)
- The acceleration of disaggregated RAN user plane network functions using eBPF/XDP with potential offloading on SmartNICs, described in D4.1 [42] and below.

The DESIRE6G WP4 RAN research focus is the CU-UP and further disaggregated DU user/data plane (termed DU-U) network functions and use of COTS CPUs and potentially SmartNIC hardware offloading. In the commercial 5G Open RAN market for the 'heavy hitter' computationally intensive functions (that are typically in the DU-U L1 High PHY FEC and CU-UP PDCP ciphering and deciphering algorithms), there are 2 main commercial directions to resolve this by either i) Plugging-in L1 High PHY PCI cards or integrated SoC offload designs or ii) leveraging the instruction set extension evolution (such as Intel AVX-512 and APX [43] and COTS Linux-based platform optimizations like eBPF/XDP. DESIRE6G WP4 is researching the latter direction.

The DU-U and CU-UP RAN functions require real-time performance, whereby not only throughput, but also bounded delay and jitter is vital. Even with real-time Kernel extensions and techniques such as CPU pinning, the worst-case time guarantees cannot be guaranteed, especially when running as cloud-native containers and virtual machines. The research therefore focusses on using 'below kernel' frameworks like eBPF/XDP [44], for 2 use-cases, illustrated below in Figure 48:

- XDP offload of CU-UP network functions to SmartNIC
- XDP programming to maximise pseudo-hard real-time performance when hosting multiple DU instances on the same DU COTS platform (with dedicated optical fronthaul NIC ports for each DU instance), for Private (NPN) Open RAN based B5G solutions.

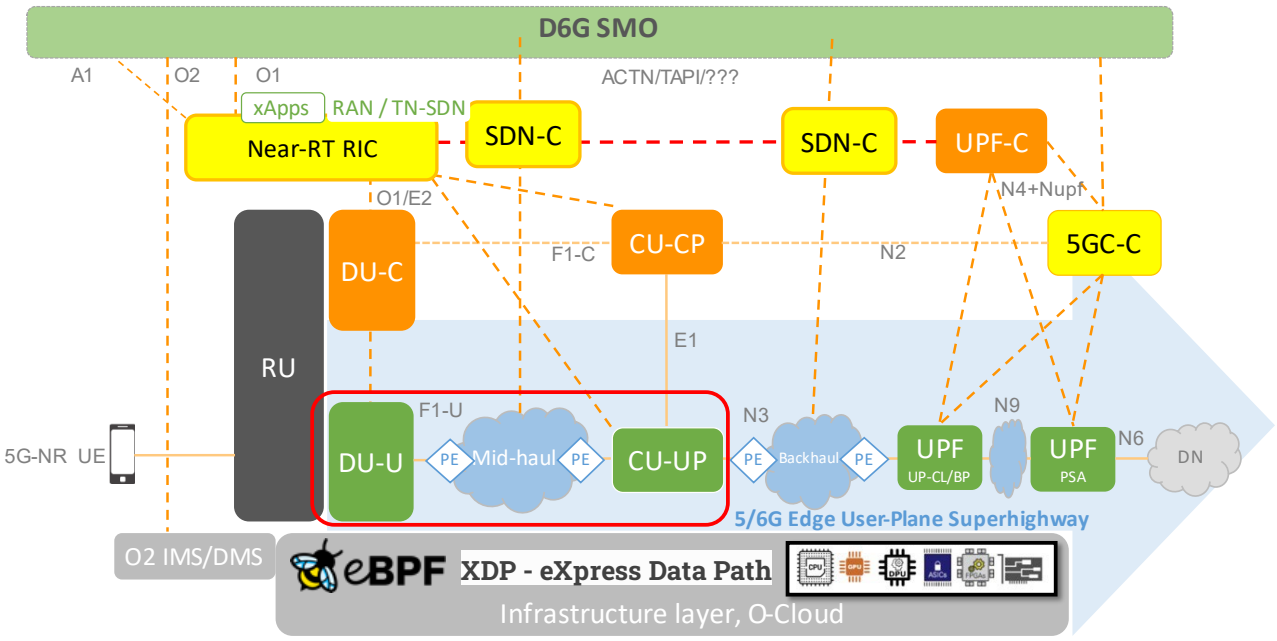

FIGURE 48: DESIRE6G RAN UTILIZING EBPF/XDP USER/DATA PLANE PROGRAMMABILITY

7.2 Core network

In the DESIRE6G system the core network is composed of infrastructure-internal components and 3GPP compliant components. The core network user plane consists of a UPF-CP that is in charge of communicating with the 3GPP CN CP elements (e.g., SMF in 5G) during events triggering the UP change (e.g., PDU Session establishment/change) while the data plane functionality is distributed to multiple network functions, including both infrastructure functions (brown) and business NFs (green), as shown in the example from Figure 49.

- UE2Service mapper: Its main role is to (based on info from UPF-CP) map the traffic from a specific user to a specific service and processing site. In downlink direction, it maps the destination IP address of incoming packets to a service id and RAN site location id, i.e., it generates a DESIRE6G header with service id and RAN site location id and the id of the next NF of UPF-DP. This corresponds to the PFCP session lookup, packet detection and forwarding actions taken in the 5G UPF.
- Buffering: it applies buffering based on Buffering Action Rules (BAR) for the inactive UEs, including the notification to the UPF-CP when a downlink packet arrives.

- QoS handler (intra-slice): it ensures the proper resource sharing between the different users using the same service. Note that, as all other UP nodes, the UPF should implement the traffic management functionality to ensure proper QoS. It may act based on the QoS Enforcement Rules (QER) received from the control plane.
- Transport adapter: it is the first (or last in downlink) component of the UPF. It performs encapsulation/decapsulation if a specific tunnelling protocol is used between the sites. A selected set of tunnelling protocols may be supported. Note that in homogeneous DESIRE6G domains it may not be needed.
- Usage reporting: it sends usage reporting information to the control plane (through UPF-CP) if required.

Note that the "UE to service mapper" (downlink) function is mandatory and a key element of the DESIRE6G system, while the rest of the UPF functionality can be added as optional "plug-in" NFs. Also note that the "UE to service mapper" function also contains the position and activity information of the UE, so it will actively communicate (through the UPF-CP) with the control plane entities handling UE location and activity states.

This decomposition and the service definition and the system-internal NF-routing functionality means that the traditional mobility tunnelling (GTP-U) is not required in our case, although it is an implementation option for the internal NF-routing functionality. The DESIRE6G system gives freedom for the operators and network developers to choose their preferred transport option (e.g., GTP, VxLAN, GRE, SRv6) and even change it "on the fly".

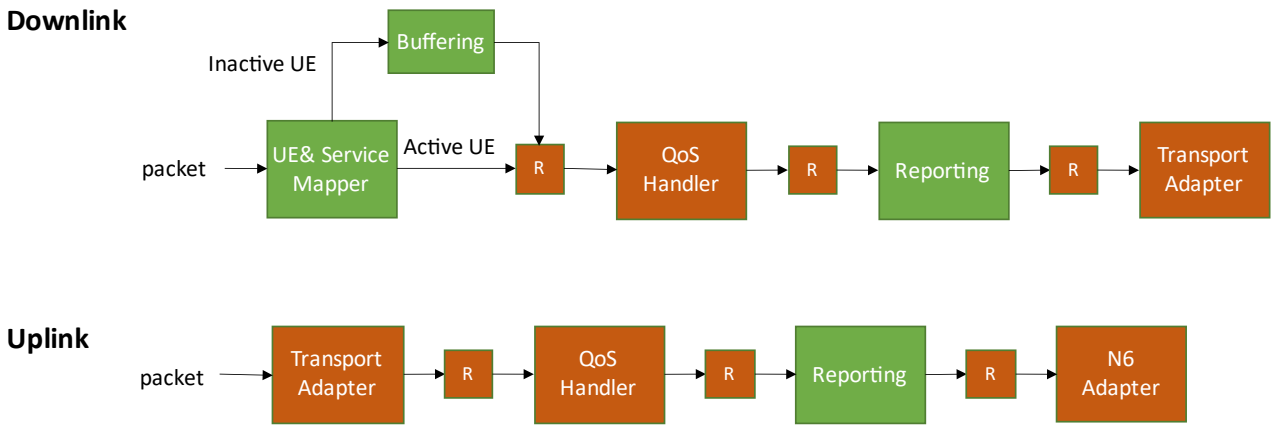

FIGURE 49: DECOMPOSED UPF – BUSINESS LOGIC VS. INFRA MODULES

Also note that in the case of end-to-end services the service mapping can be done directly at the endpoints, e.g., the UE to service mapper can be a “sidecar” functionality of the application function (AF). In this case service mapping will be a much more distributed user plane function, but since the NF-DPs and NF-CPs are separated, we can still maintain a (few) central NF-CP(s) which makes it simpler to handle the required 3GPP CP connections using the existing 3GPP interfaces.

8. Additional architecture considerations

This section contains a few additional items like possible innovations to be used also in the DESIRE6G context (e.g., MAS Security) and external, already existing or planned entities that are potential integration targets (e.g., ALTO).

8.1 MAS Security

Being security-sensitive, DESIRE6G's MAS agents are leveraging a flexible and distributed security by a permissioned DLT. Note that although the example will be for the MAS communication, the selected technology could work between any other entries communicating with each other.

Packaged in Docker containers, Ethereum DLT nodes can be easily instantiated on any available servers over the network. These nodes form a secure lattice that serves as the backbone for two security services: agent mutual remote attestation and secure key exchange. The current implementation employs proof of authority consensus among nodes for registering new nodes and creating new blocks. Authenticated smart contracts are loaded onto the DLT as transactions. Two specific smart contracts are used to orchestrate the security services. The DESIRE-6G MAS agent mutual attestation scheme eliminates hardware or kernel dependencies (e.g., TPM, IMA) but requires agent preparation through a SECaaS, generating attestation-ready, self-sufficient agents.

The procedure is described in Figure 50. Before dispatching a new agent over the network, the MLFO uses the REST API of the SECaaS and gets back deployable agents, with appended adds-on for attestation measurement and verification and a unique agent identifier. From this step onwards, the MLFO deploys the dockerized agent at its destination. As soon as the agent is bootstrapped, it will be recognized with a HTTPS post by its identifier by the remote attestation smart contract and its remote attestation will be carried out, triggering a new code block writing.

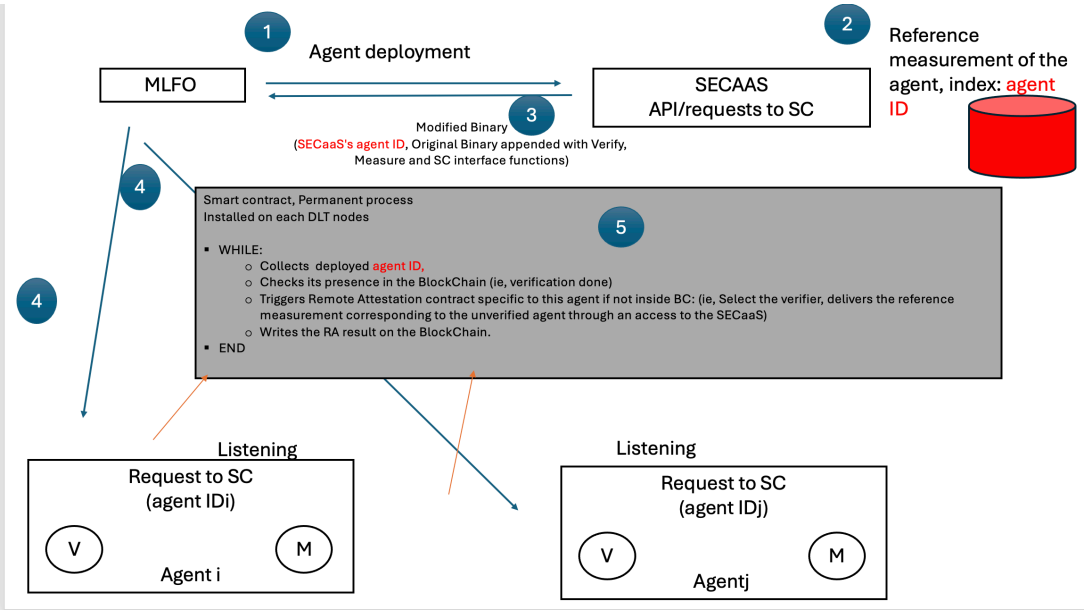

FIGURE 50: MAS AGENT SECURITY LEVERAGING DLT AND REWRITING

The Dockerized SECaaS is identified and connects to the DLT as an agent. It delivers the smart contract with the reference measurement of the agent under verification. This reference measurement is provided by the smart contract to the last verified agent, which acts as the verifier for the current remote attestation process. A second smart contract facilitates key exchange with attested agents. VxLAN connections over IPsec are created for consensus-agreed agent-to-agent links. The key exchange occurs through the DLT during the creation of these inter-agent links, minimizing latency when the link is active.

The detailed specification of the MAS security framework is provided in Deliverable 3.1 [26].

8.2 Transport network

In order to interact with the transport network, the DESIRE6G architecture considers the potential use of an external Transport SDN controller (tSDNC) supporting an Intent-Based request management NBI and the use of an external exposure module of the transport network capacities.

The first integration point of the DESIRE6G architecture with the transport network relies on the presence of a Transport SDN controller. Through the use of a Transport SDN controller, the SMO can dynamically request connectivity services to the transport network, enabling end to end automation and programmability, thus going in line with the rest of the components of the DESIRE6G architecture. For that purpose, the Transport SDN controller has to expose a services NBI to receive connectivity requests from higher network domains (the SMO in this case), thus providing site to site connectivity. In case the

transport network includes some network elements that are P4-programmable, this Transport SDN controller may have a P4 device management module in its SBI to dynamically manage the data plane of those network elements.

The transport SDN controller considered in the DESIRE6G architecture is connected to the SMO through and intent-based networking (IBN) interface for the purpose of managing the transport network (see Figure 51). This IBN module, acts as a translator connecting the DESIRE6G architecture to the transport subdomain, being technological agnostic and therefore being also independent of the SDN controller managing the transport network. This gives the SMO ability to adapt to different transport technologies, relying on the flexibility inherited by the IBN paradigm. One such SDN controller could be Teraflow, if extended to support the Intent Based Network interface used by the SMO. The SMO interface to the transport subdomain follows an RDF structure defined in the TM Forum [37].

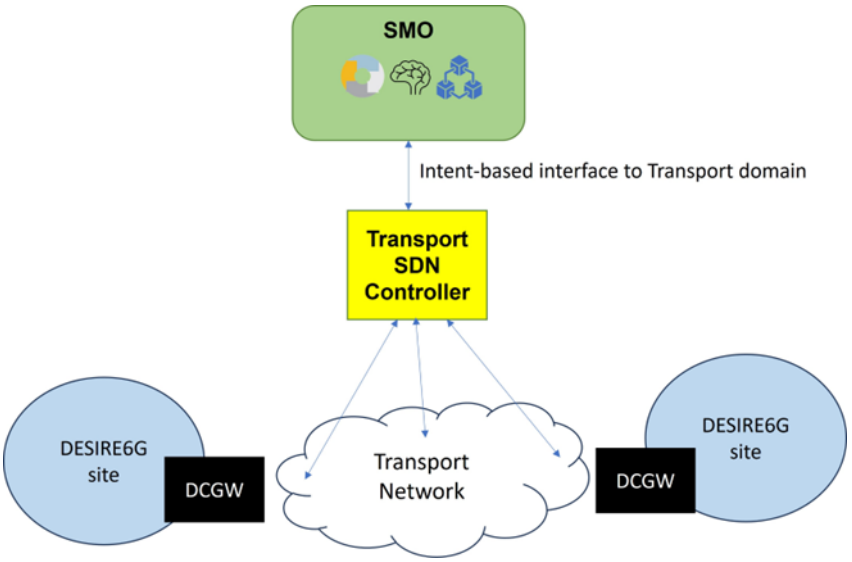

FIGURE 51: HIGH-LEVEL VIEW WITH THE SMO-SDNC INTERFACE

On the other hand, in order to provide visibility of the managed topology in the transport network, it is necessary to have an external exposure module present in the transport network. To perform this task, transport network operators can rely on the IETF Application-Layer Traffic Optimization (ALTO) protocol (RFC7285 [45]). This protocol is designed to be able to receive information from different sources, being the transport network topology and metrics (i.e. hop count, bandwidth and latency) the main one for which it is designed. This feature allows it to be a module that fits with the DESIRE6G architecture, being able to read information from the transport network through BGP speakers, accessing to the devices and metrics programed and exposed by this technology.

This exposure module would be also integrated with the SMO, providing him visibility of the state and capabilities of the transport network via the Topology Component. This relationship is represented in Figure 52 and would be a parallel interface to the one with the Transport SDN controller exposed previously in this section, complementing it.

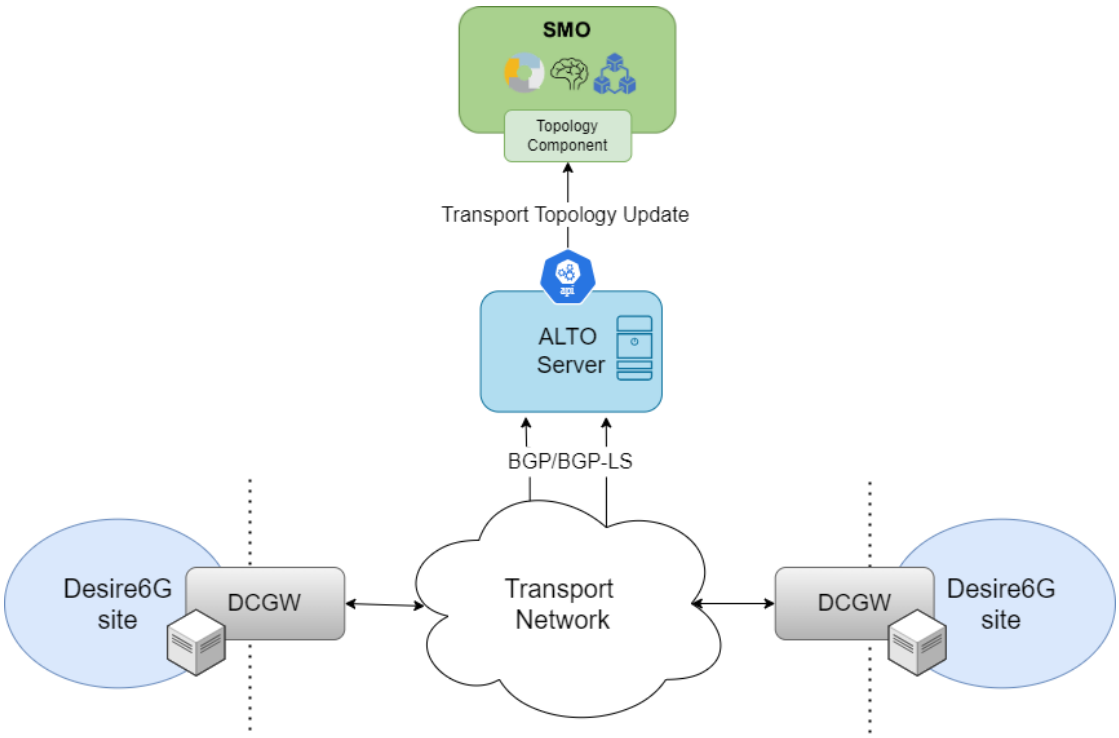

FIGURE 52: THE ALTO-BASED INTERFACE FOR GETTING TOPOLOGY INFORMATION

8.3 TSN and DetNet extensions

In order to comply with the latency and reliability requirements of the DESIRE6G use cases, the transport network connecting the DESIRE6G sites may need to support appropriate protocols such as TSN or Detnet that target this kind of requirements. In the case of a Service Provider network spanning the geography of large areas of a country, DetNet is the more suited technology because of its Layer 3 nature, in line with the typical Service Provider networks that are Layer 3 based.

Extending the DESIRE6G programmable data plane layer to the transport network faces still different challenges. Service Provider transport networks typically encompass thousands of nodes with high switching capacity (higher as you aggregate traffic in the centralized transport locations). Transport networks are typically multiservice in nature, being the transport of mobile traffic just one of the services they support, and as a result the number of communication protocols and features to be supported is very high.

All of the previous make the case for having the transport network as a whole remain an underlay supporting the traffic of the mobile NFs deployed in the DESIRE6G sites. As mentioned in the previous section that transport underlay would be controlled by means of a Transport SDN controller and its capabilities exposed by means of an ALTO server.

In spite of that underlay consideration of the transport network, the support of programmable HW targets in some transport nodes may enable the support of new protocols, specially at the provider edge nodes, in a much more agile fashion than what has been the norm for decades in the telecommunications industry. For that reason, the DESIRE6G architecture considers that some transport nodes may already be programmable, if the chipsets with the required capacity are available.

In particular, the DESITRE6G architecture considers that DetNet support to connect the DESIRE6G sites could be partially included in the transport network by means of programmable nodes. DetNet opens the possibility of transporting unicast or multicast flows of real-time applications that require an extremely low data loss rate. DetNet emerges as an extension of Time-Sensitive Networking (TSN) designed for Layer 3. It operates at the IP layer and can connect to subnetworks including MPLS Traffic Engineering (TE), IEEE TSN, and Optical Transport Network (OTN). It is expected that the transmission reliability of a 32-byte packet will be at least 99.999%, with limited latency, such as < 1 millisecond, within a domain.

DetNet has its own stack and is composed of two layers: the Service sub-layer and the Forwarding sub-layer, as we can show in Figure 53. DetNet functionalities are distributed across one of these two layers; for example, service protection would be in the Service sub-layer, while explicit routes and resource allocation to DetNet flows belong to the Forwarding sub-layer. DetNet nodes may present only one of these sub-layers, both, or neither, depending on the functionalities they incorporate.

Furthermore, a deterministic network is composed of End-systems, Edge nodes, Relay nodes, and Transit nodes, as shown in Figure 53.

- End-system: An end-system may or may not have DetNet functionalities, which would imply having both, one, or none of the layers of the stack. In the event of not containing any, service proxies are necessary.
- DetNet Edge, Relay, and Transit Nodes: Both the Edge nodes and the Relay nodes are DetNet aware because they both contain Service and Forwarding sub-layers. Additionally, the Edge

nodes provide proxy service. In contrast, Transit nodes only need to be aware of the DetNet Forwarding sub-layer.

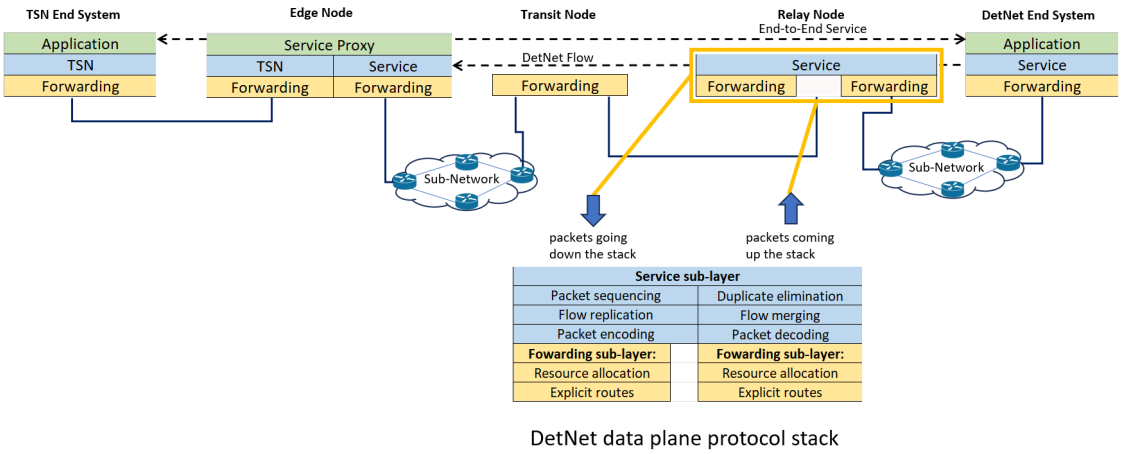

FIGURE 53: THE DETNET ARCHITECTURE

The use of DetNet in the DESIRE6G architecture would allow to provide connectivity services between DESIRE6G sites, as shown in Figure 54, according to the KPIs of very low latency or high reliability applications. The DESIRE6G project has already made several contributions to the IETF in that respect [38], [39], [40], [41].

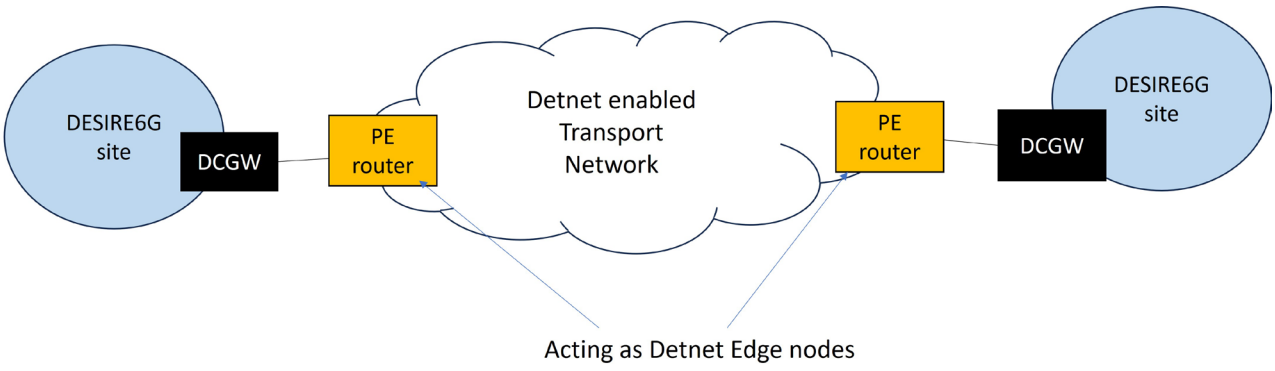

FIGURE 54: USING DETNET FOR TRANSPORT BETWEEN DESIRE6G SITES

Within the functionalities offered by DetNet, Packet Replication, Elimination, and Ordering Functions (PREOF) at the DetNet Service sub-layer, is aimed to enhance network reliability. PREOF encompasses four capabilities:

1. Packets in a DetNet composite flow contain a sequence number or timestamp to carry sequencing information. Typically, this timestamping is done only once at the source node.
2. Packets are replicated and distributed across multiple paths to the same destination (Packet Replication Function).
3. Based on sequencing information and the history of received packets, duplicate packets are eliminated (Packet Elimination Function). The output of PEF is a single packet.
4. Additionally, through sequencing information, packets from a DetNet flow that are received out of order can be reordered (Packet Ordering Function).

9. Summary and conclusions

This deliverable has finalized the definition of the DESIRE6G architecture, showing the way how the different innovations across the multiple layers of the system will interact and thus create a dynamic, service-centric system. Section 2 showed the "big picture", section 3-5 discussed the details of the different layers, while section 6 talked about system processes for creating, deploying and optimizing services. It also contains two functions to show how the described service-centric architecture can interact with basic mobile network functionality.

The current document is a basis for the future integration and implementation work. From this point, the project will continue to work on implementing the necessary software modules in each layers described here. Further documentation of the orchestration and optimization layers will be covered by D3.2 and D3.3, "Interim / Final report on the intelligent and secure management, orchestration, and control", while the programmable data plane innovations including monitoring and hardware acceleration work will be documented in D4.2 and D4.3, "Interim / Final report on the DESIRE6G unified programmable data plane layer".

Besides the development work integration and proof-of-concept prototype building has been already started, but the bulk of the tasks are yet to come. Both the main demonstrators and the smaller, local PoCs will use the architecture components described in this document and will be reported in D5.2 and D5.3, "Report / Final report on evaluation results and initial proof-of-concept demonstrations".

Work with the architecture is closed, but not finished with the submission of the current document. The ideas and innovations presented here will have to be fed into the European 6G ecosystem. Some initial steps have been already made, e.g., presenting the main ideas to the SNS JU Architecture WG or to the Hexa-X II. flagship project, but here more future efforts are expected.

10. References

[1] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," IEEE Communications Surveys & Tutorials, vol. 25, no. 2, pp. 1376-1411, Jan. 2023.

[2] 3GPP TS 28.530, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and orchestration; Concepts, use cases and requirements."

[3] ETSI GS ZSM 003, "Zero-touch network and Service Management (ZSM); End-to-end management and orchestration of network slicing."

[4] C. Y. Chang et al., "Performance isolation for network slices in industry 4.0: The 5growth approach," IEEE Access, vol. 9, pp. 166990-167003, 2021

[5] ITU-R, "IMT towards 2030 and beyond," [Online]. Available: https://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2030/Pages/default.aspx.

[6] O-RAN Alliance, "O-RAN nGRG Research Reports," [Online]. Available: https://www.o-ran.org/research-reports.

[7] 3GPP, "3GPP Stage 1 workshop on IMT 2030 use cases," 2024. [Online]. Available: https://www.3gpp.org/ftp/workshop/2024-05-08_3GPP_Stage1_IMT2030_UC_WS/Docs/SWS-240025.zip.

[8] SNS, "The European view on 6G use cases," 2024. [Online]. Available: https://www.3gpp.org/ftp/workshop/2024-05-08_3GPP_Stage1_IMT2030_UC_WS/Docs/SWS-240018.zip.

[9] ETSI, "ETSI Software Development Group TeraFlowSDN," [Online]. Available: https://tfs.etsi.org/.

[10] ETSI, "Zero touch network & Service Management (ZSM)," [Online]. Available: https://www.etsi.org/technologies/zero-touch-network-service-management.

[11] ETSI, "ETSI Software Development Group for OpenSlice," [Online]. Available: https://osl.etsi.org/.

[12] tmforum, "TM Forum's Open API program," [Online]. Available: https://www.tmforum.org/oda/open-apis/.

[13] O-RAN Alliance, "69 New or Updated O-RAN Technical Documents Released since November 2023," 22 03 2024. [Online]. Available: https://www.o-ran.org/blog/69-new-or-updated-o-ran-technical-documents-released-since-november-2023.

[14] 3GPP, "3GPP TS 29.520, 5G System; Network Data Analytics Services; Stage 3, v18.6.0," 24 06 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3355.

[15] MARSAL Project, "MARSAL H.2020 Project Public Deliverables," [Online]. Available: https://www.marsalproject.eu/public-deliverables/.

[16] O-RAN Alliance, "O-RAN nGRG: next Generation Research Group," [Online]. Available: https://public.o-ran.org/display/NGRG/Introduction.

[17] O-RAN Alliance nGRG, "Research Report on Native and Cross-domain AI: State of the art and future outlook," 14 12 2023. [Online]. Available: https://mediastorage.o-ran.org/ngrg-rr/nGRG-RR-2023-03-Research-Report-on-Native-and-Cross-domain-AI-v1_1.pdf.

[18] 3GPP, "3GPP TS 28.533 Management and orchestration; Architecture framework, v18.1.0," 05 04 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3416.

[19] 3GPP, "3GPP TS 23.288: Architecture enhancements for 5G System (5GS) to support network data analytics services, v18.5.0," 27 03 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3579.

[20] 3GPP, "3GPP TS 38.300: NR; NR and NG-RAN Overall description; Stage-2, v18.1.0," 03 04 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3191.

[21] T. Faisal, J. A. O. Lucena, D. R. Lopez, C. Wang, and M. Dohler, "How to Design Autonomous Service Level Agreements for 6G," IEEE Communications Magazine, vol. 61, no. 3, pp. 80-85, Mar. 2023, doi: 10.1109/MCOM.001.2200131.

[22] N. Gritli, F. Khendek, and M. Toeroe, "Decomposition and Propagation of Intents for Network Slice Design," 2021 IEEE 4th 5G World Forum (5GWF), Montreal, QC, Canada, 2021, pp. 165-170, doi: 10.1109/5GWF52925.2021.00036.

[23] ETSI GS ZSM 001, "Zero-touch network and Service Management (ZSM); Requirements based on documented scenarios: ETSI GS ZSM 001 Vl.1.1", 2019.

[24] 3GPP TS 28.812, "Technical Specification Group Services and System Aspects; Telecommunication management; Study on scenarios for Intent driven management services for mobile networks (Release 16)", March 2020.

[25] "Autonomous networks with multi-layer, intent-based operation", [Online]. Available: https://www.ericsson.com.

[26] DESIRE6G deliverable 3.1, "Initial report on the intelligent and secure management, orchestration, and control platform," [Online]. Available: https://zenodo.org/records/10356033.

[27] Apache Kafka, [Online]. Available: https://kafka.apache.org/,

[28] Y. Kim, D. Suh, and S. Pack, "Selective in-band network telemetry for overhead reduction," in 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), IEEE, 2018, pp. 1-3.

[29] S. Tang, J. Kong, B. Niu, and Z. Zhu, "Programmable Multilayer INT: An Enabler for AI-Assisted Network Automation," IEEE Communications Magazine, vol. 58, no. 1, pp. 26-32, Jan. 2020.

[30] D. Suh, S. Jang, S. Han, S. Pack, and X. Wang, "Flexible sampling-based in-band network telemetry in programmable data plane," ICT Express, vol. 6, no. 1, pp. 62-65, 2020.

[31] S. Tang, D. Li, B. Niu, J. Peng, and Z. Zhu, "Sel-INT: A runtime-programmable selective in-band network telemetry system," IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 708-721, 2019.

[32] G. Simsek, D. Ergenç, and E. Onur, "Efficient network monitoring via in-band telemetry," in 2021 17th International Conference on the Design of Reliable Communication Networks (DRCN), IEEE, 2021, pp. 1-6.

[33] D. Mo, Z. Liu, D. Chen, and D. Gao, "C-INT: An Efficient Cluster Based In-Band Network Telemetry," in 2021 4th International Conference on Hot Information-Centric Networking (HotICN), IEEE, 2021, pp. 129-134.

[34] R. Ben Basat et al., "PINT: Probabilistic in-band network telemetry," in Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 662-680, 2020.

[35] E. Song et al., "INT-label: Lightweight In-band Network-Wide Telemetry via Interval-based Distributed Labelling," in IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1-10, doi: 10.1109/INFOCOM42981.2021.9488799.

[36] A. Karaagac, E. De Poorter, and J. Hoebeke, "In-Band Network Telemetry in Industrial Wireless Sensor Networks," IEEE Transactions on Network and Service Management 17, 1 (2020), 517–531. https://doi.org/10.1109/TNSM.2019.2949509

[37] TM Forum, "Intent Common Model – Intent Expression v3.2.0 (TR290A)," [Online]. Available: https://www.tmforum.org/resources/technical-report/tr290a-intent-common-model-intent-expression-v3-2-0/.

[38] A. Malis, X. Geng, M. Chen, F. Qin, B. Varga, and C. Bernardos, "Deterministic Networking (DetNet) Controller Plane Framework," [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-detnet-controller-plane-framework/06/.

[39] C. J. Bernardos and A. Mourad, "RAW multidomain extensions," [Online]. Available: https://datatracker.ietf.org/doc/draft-bernardos-detnet-raw-multidomain/02/.

[40] L. M. Contreras, M. Boucadair, D. Lopez, and C. J. Bernardos, "An Evolution of Cooperating Layered Architecture for SDN (CLAS) for Compute and Data Awareness," [Online]. Available: https://datatracker.ietf.org/doc/draft-contreras-coinrg-clas-evolution/02.

[41] R. S. Sofia, P. Mendes, and C. J. Bernardos, "Requirements for Reliable Wireless Industrial Services," [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-detnet-raw-industrial-req/00.

[42] DESIRE6G deliverable 4.1, "Initial report on the DESIRE6G unified programmable data plane layer," [Online]. Available: https://zenodo.org/records/10356108.

[43] Intel, “vRAN Boost in 4th Gen Xeon CPUs,” [Online]. Available: https://cdrdv2-public.intel.com/781303/2023MWC-vRAN-Fact-Sheet.pdf.

[44] “The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel,” [Online]. Available: https://github.com/xdp-project/xdp-paper/blob/master/xdp-the-express-data-path.pdf.

[45] J. Seedorf and E. Berger, “Application-Layer Traffic Optimization (ALTO) Protocol,” RFC 7285, Sep. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7285.

[46] The P4.org Applications Working Group, ”In-band network telemetry data-plane specification 2.1,” [Online]. Available: https://p4.org/p4-spec/docs/INT_v2_1.pdf

[47] K. Papadopoulos, P. Papadimitriou, and C. Papagianni, ”Deterministic and Probabilistic P4-Enabled Lightweight In-Band Network Telemetry,” IEEE Transactions on Network and Service Management, Aug. 3, 2023.

[48] Patent: ”Packet Handling in Heterogeneous Networks with different Data Plane Network Function Implementations”, authors: A. Mihály, G. Pongrácz, P. Vörös, S. Laki, status: application, reference no.: P109374WO01 (2024-01-31)

[49] Patent: ”Load Balancing with Instance Scaling”, authors: A. Mihály, G. Pongrácz, P. Vörös, S. Laki, status: application, reference no.: P109937WO01 (2024-02-29)

[50] Patent: ”Hardware-based load balancer with network function scaling”, authors: A. Mihály, G. Pongrácz, P. Vörös, S. Laki, status: application, reference no.: P109939WO01 (2024-02-29)

11. Annex I: An example local network service descriptor

An example local network service descriptor (local NSD) SMO sends to IML at a specific site. The descriptor lists network and application functions, management networks, site connections, forwarding graphs for both uplink and downlink directions, resource sharing policies and QoS classes.

```
local-nsd:
  ## local-nsd refers to a network service graph whose subgraph is described
  info:
    ns-instance-id: "55" # same as the 16bit long identifier used in D6G Header
    ns:
      name: "Internet Access"
      id: "418420e3-e2a8-4338-bc8c-be685548bad2" # refers to the ns in the
service catalog
      vendor: "D6G"
      descriptor-version: "1.0"
    site-id: "d6g-001"
  data-network:
    addressing-type: "ipv4"
    prefix: "10.23.0.0"
    prefix-length: "16"
  mgmt-networks:
    - mgmt-network-id: "shared-mgmt-net"
      addressing-type: "ipv4"
      prefix: "10.1.0.0"
      prefix-length: "16"
      local-range: "10.1.1.1-10.1.2.254"
    - mgmt-network-id: "mas-kafka-bus"
      addressing-type: "ipv4"
      prefix: "10.121.0.0"
      prefix-length: "16"
      local-range: "10.121.10.1-10.121.20.254"
  network-functions:
    - nf-instance-id: "flowcl-i01"
      nf-instance-d6g-id: "101"
      nf-id: "flowcl"
      nf-name: "Flow Classifier"
      nf-version: "v1.0"
      nf-mgmt-network: "shared-mgmt-net" # provides access to higher level
controllers
    - nf-instance-id: "firewall-i01"
      nf-instance-d6g-id: "102"
      nf-id: "firewall"
      nf-name: "ACL Firewall"
```

```
      nf-version: "v1.0"
      nf-mgmt-network: "shared-mgmt-net"
  application-functions:
    - af-instance-id: "lws-i01"
      af-instance-d6g-id: "103"
      af-id: "localwebserver"
      af-name: "Local web server for as an example service"
      af-version: "1.0"
      af-mgmt-network: "shared-mgmt-net" # optional
  mas-agent:
    instance-id: "mas0012"
    id: "smas001"
    name: "service_mas"
    version: "v1.0"
    mas-mgmt-network: "mas-kafka-bus"
  site-connections: # External connections
      - site-connection-id: "sc1-sc2"
        site-id-ref: "d6g-002"
        tunnel-id: "444" # tunnels are preconfigured by SDNC - Transport adapter
implements it.
      - site-connection-id: "sc1-sc3"
        site-id-ref: "d6g-003"
        tunnel-id: "321"
      - site-connection-id: "sc1-sc4"
        site-id-ref: "d6g-004"
        tunnel-id: "123"
  resource-sharing-policies:
    - rs-policy-id: "1"
      rs-policy-name: "gold"
      guaranteed-bw: "1Mbps"
      guaranteed-weight: "2"
      best-effort-segments:
        - best-effort-segment-index: 1
          best-effort-bw: "100Mbps"
          best-effort-weight: "2"
        - best-effort-segment-index: 2
          best-effort-bw: "1000Mbps"
          best-effort-weight: "1"
    - rs-policy-id: "2"
      rs-policy-name: "silver"
      guaranteed-bw: "1Mbps"
      guaranteed-weight: "2"
      best-effort-segments:
        - best-effort-segment-index: 1
          best-effort-bw: "1000Mbps"
```

```
          best-effort-weight: "1"
  qos-groups:
    - qos-group-index: 1
      resource-sharing-policy-id: "1" # gold
      latency-preference-class: "on-demand-low" # handling according to L4S flag,
options: "ultra-low" (<1ms), "low" (1-2ms), "normal" (~20ms), "on-demand-low"
(low/normal according to L4S flag)
    - qos-group-index: 2
      rule: "default"
      resource-sharing-policy-id: "2" # silver
      latency-policy-class: "normal"
  forwarding_graphs:
      - member-graph-index: 1 # list of subgraphs/forwarding segments implemented
by the given D6G site
        graph-name: "uplink"
        links:
          - link-id: "input-1"
            connection-points:
            - member-connection-point-index: 1
              member-if-id-ref: "external-site" # external-site (tr adapter
config determines the site and tunnel)
            - member-connection-point-index: 2
              member-if-id-ref: "flowcl-i01:port-1"
          - link-id: "towebserver"
            connection-points:
            - member-connection-point-index: 1
              member-if-id-ref: "flowcl-i01:port-2"
            - member-connection-point-index: 2
              member-if-id-ref: "lws-i01:eth0"
          - link-id: "toFW"
            connection-points:
            - member-connection-point-index: 1
              member-if-id-ref: "flowcl:port-3"
            - member-connection-point-index: 2
              member-if-id-ref: "firewall-i01:port-1"
          - link-id: "toNextSite"
            connection-points:
            - member-connection-point-index: 1
              member-if-id-ref: "firewall-i01:port-2"
            - member-connection-point-index: 2
              member-if-id-ref: "external-site:next-nf-id=431" # forwarding to
another D6G site with setting the next-nf-id
        site-delay-budget: "5ms"
      - member-graph-index: 2 # list of subgraphs/forwarding segments implemented
by the given D6G site
```

```
        graph-name: "downlink"
        links:
          - link-id: "input-1"
            connection-points:
            - member-connection-point-index: 1
              member-if-id-ref: "external-site"
            - member-connection-point-index: 2
              member-if-id-ref: "external-site" # forwarding to another site
without modifying the next-nf-id in the D6G header
          - link-id: "fromwebserver-1"
            connection-points:
            - member-connection-point-index: 1
              member-if-id-ref: "lws-i01:eth0"
            - member-connection-point-index: 2
              member-if-id-ref: "external-site:next-nf-id=202"
        site-delay-budget: "2ms"
```