FUDGE-5G

FUlly DisinteGrated private nEtworks
for 5G verticals

Deliverable 1.2

FUDGE-5G Platform Architecture Components and Interface

Version 1.0

Work Package 1

Editor	Sebastian Robitzsch
Status	PU
Delivery date	August 2021

Partners

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957242

Versioning and Contribution History

#	Description	Contributors
0.1	Table of contents	IDE
0.2	Section assignments	all
0.3	Section 2 Contributions SCP, SFV and architecture contributions	CMC, IDE
0.4	Input for Section 3.1 and 3.2 in relation to the routing component of the FUDGE-5G platform	HWDU
0.5	Further SFV contributions, gPTP for TSN, 5G multicast, OAM procedures for FOKUS and CMC core, VAO description	IDE, 5CMM, FOKUS, UBI, CMC
0.6	Orchestration architecture update	IDE
0.7	IP Multicast in 5G	UPV
0.8	Opportunistic Multicast, VAO description update	IDE, UBI
0.9	Added ATH O&M, Revised FOKUS O&M, O2M	ATH, FOKUS, O2M
0.10	Slicing to enable PNI-NPN, 5CMM reviewed 5GLAN and TSN sections, OAM for verticals, VAO description, updated resource scheduling and user vs control plane consideration	ATH, UBI, HWDU
0.11	SCP background and proposition within FUDGE-5G, Slicing of 5GC NFs, Service routing on the user plane	IDE
0.12	VAO description, OAM for AFs	UBI, ONE
0.13	Monitoring for Enterprise Services, Completing 5GLAN section	UBI, CMC
0.14	Revised monitoring section, Revised orchestration section, completed architecture and routing section	UBI, IDE
0.15	End-to-end slicing, Executive Summary	THA, IDE
0.16	Revised SCP section, addressed reviewer's comments, adding Introduction and Conclusion	FHG, UPV, IDE
1.0	Generating final version	IDE

List of Authors and Reviewers

Authors	Partner

Sebastian Robitzsch, Chathura Sarathchandra	IDE
Jose Costa-Requena	CMC
Zoran Despotovic	HWDU
Andrés Meseguer, Manuel Fuentes	5CMM
Pousali Chakraborty	FOKUS
Thanos Xirofotos	UBITECH
Daniele Munaretto, Marco Centenaro, Nicola di Pietro	ATH
Peter Sanders	O2M
Carlos Barjau, David Gómez-Barquero, Josep Ribes, Borja Iñesta	UPV
Filippo Rebecchi	THA
Luís Cordeiro, André Gomes, António Borges	ONE

Reviewer	Partner
Steve Appleby	British Telecom (BT)

Acronyms

5GC	5G Core
AMF	Attachment and Mobility Function
ARP	Allocation and Retention Priority
CADS	Compute-Aware Distributed Scheduling
CID	Content Identifier
CLA	Cross Layer Analytics
COTS	Commodity-of-the-Shelf
CNF	Cloud Native Network Function
CP	Control Plane
FQDN	Fully Qualified Domain Name
FID	Forwarding Identifier
gNB	Next Generation NodeB
HPLMN	Home Public Land Mobile Network
IETF	Internet Engineering Task Force
ITIL	Information Technology Infrastructure Technology
LCM	Lifecycle Management
NbR	Name-based Routing
NWDAF	Network Data Analytics Function
NF	Network Function
PLMN	Public Land Mobile Network

RFC	Request for Comments
SBA	Service-based Architecture
SC	Service Chain
SCC	Service Chain Controller
SCP	Service Communication Proxy
SF	Service Function
SFC	Service Function Chaining
SFE	Service Function Endpoint
SFEC	Service Function Endpoint Controller
SFV	Service Function Virtualisation
SFVO	Service Function Virtualisation Orchestrator
SH	Service Host
SLA	Service Level Agreement
SLAM	Service Level Agreement Manager
SMF	Session Management Function
TLS	Transport Layer Security
TSN	Time Sensitive Networking
UPF	User Plane Function
VAO	Vertical Application Orchestrator
VN	Virtual Network
VPLMN	Visiting Public Land Mobile Network

Executive Summary

This deliverable of the FUDGE-5G project marks the first release of the platform architecture specification. FUDGE-5G innovates in the area of mobile telecommunication systems by evolving the concepts around Service-based Architecture by introducing a Platform-as-a-Service offering composed of service routing, orchestration (provisioning and lifecycle management), telemetry and slicing and treating 5G Cores and vertical application as Enterprise Applications. Additionally, FUDGE-5G innovates on Enterprise Applications for mobile telecommunication systems such as 5GLAN, Time Sensitive Networking or 5G Multicast.

Table of Contents

1 Introduction

FUDGE-5G aims to devise, assess and demonstrate a conceptually novel and forward-looking cloud-native, unified and secured service-based 5G architecture, solutions and systems for Non-Public Networks. To achieve this range of objectives an overall system architecture, its components and interfaces has been designed in the first twelve months that builds the required foundations.

With 3GPP Release 18 officially started at the time of finishing this deliverable, it can be safely stated that the work presented in this deliverable is based on Release 17 with technologies such Service-based Architecture, Time Sensitive Networking, 5G LAN or 5G Multicast. Combined with the tremendous pace of cloud technologies – such as Kubernetes – entering the telco edge, the time is ripe for an evolved take on a Service-based Architecture platform for mobile telecommunication systems targeting Non-Public Networks.

A key advantage of NPN lies in its applicability which is rather niche when considering the limited number of services and use cases it has to cover. Much more fine-tuned mobile telecommunication networks are required to meet the demand and expectations from the verticals that seek to offer a 5G system that delivers their service. Whether this means to remove unnecessary 5G Core functionality like sessions monitoring, charging and OSS/BSS integrations for gNB management, or tighter integration of 5G Core functionality with the service offering of a vertical like a globally operating organisation offering Business-to-Business solutions.

D1.1 of FUDGE-5G, describing the use cases and the validation framework, already presented the list of key innovations of the project that will be ultimately trialled across the five use cases. In comparison to that, Section 2 of this deliverable provides a deeper dive into the technologies used to define an evolved Service-based Architecture and its components. This lays the foundation of the FUDGE-5G system architecture, presented in Section 3 covering the actual high-level architecture and then gradually walking through its components and functionality covering service routing, orchestration, telemetry and slicing. The deliverable is concluded in Section 4.

2 Technology Background

2.1 Service-based Architecture

With 3GPP's Release 15 [3GP18] a paradigm shift in the system architecture was introduced on how 5G Core (5GC) Network Functions (NFs) communicate with each other. In pre-Release 15 systems, all 5GC NF instances had a strict 1:1 relationship among each other. With the advances of cloud solutions that can scale on demand both vertically (change properties of instances) and horizontally (change number of instances), Release 15 adopted a Service-based Architecture (SBA) for their 5G Core with the following changes:

- Introduced the concept of consumer (endpoint/clients) and producers (service endpoint/servers) without strict requirements on which consumer is allowed to communicate with which producer.
- Introduced Service-based Interfaces (SBI) for the majority of 5GC NFs moving to HTTP/2 as the application layer protocol and JSON-encoded payload.

In Release 16 [3GP19b] another key component was added to the 5G system architecture, i.e. the Service Communication Proxy (SCP). The deployment of an SCP is optional and the SCP does not expose a 5GC service itself. Instead, is used for "*indirect* communication between NFs and NF services". If no SCP is deployed consumers and producers communicate with each other without an SCP and Release 16 refers to that as a "*direct* communication".

3GPP's Release 16 allows three deployment options for an SCP [3GP19b]:

1. **SCP based on a service mesh:** A Service Mesh provides a proxy functionality that is deployed with each microservice (e.g. as a sidecar [ENG18]) and helps to route communication requests to the destination microservice proxy using the optimal route. The combination of all these proxies is called a service mesh. The determination of the targeted endpoint can be done on a global view or based on local available metrics.
2. **SCP based on independent deployment units:** This option enables a deployment unit of an SCP which can internally make use of microservices and allows therefore to be independent of the message forwarding platform. The SCP agents implement the HTTP intermediaries between service consumers and service producers. The SCP agents are controlled by the SCP controller. The SCP itself is not a service producer. It is acting as HTTP proxy which registers services on behalf of the producers in NRF. Since the SCP is acting in proxy mode, there is a need to explicitly address the SCP within the 5GC functionality for leveraging the SCP's functionality, which introduces a complexity and thus management overhead.

3. **SCP based on Name-based Routing:** This option utilises a particular information-centric networking (ICN) flavour and operates straight on top of Layer 2 using path-based forwarding identifiers (Bloomfilters) [TRO19]. A Service Router (SR) terminates the underlying transport session and uses the given target FQDN to find the most appropriate service endpoint that could serve the request. The logic translating the IP world to ICN and vice versa at the edge is implemented in the SR component. The SR acts transparently on application level. Hence, no further configuration or complexity is added to the business logic components which perform the request- and response-based transactions.

All three SCP deployment options must support the same communication design pattern for consumers. Whenever a consumer aims to communicate with a producer, the NRF provided the information if an SCP is available, which one to be used and how to address it. The addressing is either an FQDN or an IP address. The selection of the NF service (producer), the producer set (sub-set of producer instances from the pool of all producer instances of the same type) or the actual producer instance is executed by the NRF. However, Release 16 allows two modes of operation for the SCP, i.e.:

- SCP without delegated discovery
- SCP with delegated discovery

The two modes determine if the discovery of producer instances through the NRF is done by the consumer prior to the communication to the SCP (without delegated discovery) or by the SCP itself on behalf of the consumer to the NRF (with delegated discovery).

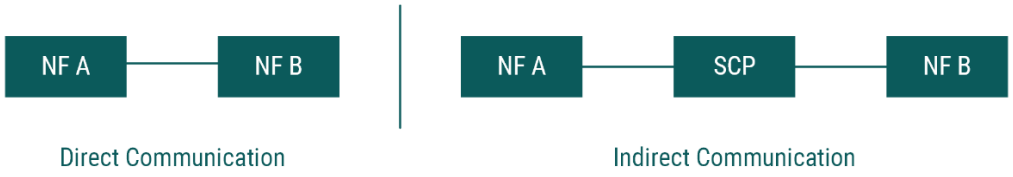

Figure 2.1: Direct and indirect commuinication options based on Release 16 [3GP19b]

For roaming scenarios, Release 17 states that "*an SCP can be used for indirect communication between NFs and NF services within the Visiting PLMN (VPLMN), within the Home PLMN (HPLMN), or within both VPLMN and HPLMN*". In other words, not for an indirect communicate between VPLMN and HPLMN. Also, two or more SCPs can be deployed in the same PLMN and multiple SCP can be used to route traffic between consumer and producers. In that case, an SCP discovers and selects the "next hop SCP" by querying the NRF or through a pre-configured set of information.

3GPP also allows various modes of the communication affinity between consumer and producer instances. While the standard still allows NRFs to enforce a long-lived binding between instances, it also supports the binding to a specific set of producer instances or even leave it entirely to the SCP to find the most appropriate one based on NRF and

consumer and producer indications. This large degree of freedom in adopting cloud concepts poses the challenge on the possibility for service routing and service provisioning concepts as a platform offering to 5GCs.

2.1.1 Relationship to FUDGE-5G

The adoption of cloud principles manifest in the Service-based Architecture principles outlined in TS23.501 since Release 15 form the basis of the FUDGE-5G platform with respect to offering routing, orchestration and monitoring of 5GC NF services in a Platform-as-a-Service (PaaS) manner. Furthermore, SBA and the concept of an SCP in particular is of key interest to the project allowing the decompostion of NFs into a set of microservices and offer their orchestration, monitoring for lifecycle management purposes and indirect communication as part of the FUDE-5G platform and in a unified fashion across all FUDGE-5G use cases.

As FUDGE-5G focuses on technological and economic benefits of utilising the innovations in an NPN setting, the project has the unique opportunity to demonstrate the benefit of operating in an NPN setting where the system follows the standard but is fine-tuned and optimised for the actual use case where it is applied to.

2.2 5GLAN

Local Area Networks (LAN) are the pillar of private networks, characterized by the high throughput, customization, low latency, and isolation against public networks. 5GLAN emulates LAN capabilities required for connecting 5G networks to fixed industrial infrastructure.

5GLAN is therefore an essential technology that will be embraced for offering private communication using IP and/or non-IP, and to seamlessly integrate 5G with fixed and wireless (WiFi) LAN. Furthermore, 5GLAN can serve as basis for more innovative technologies. The existing 5GLAN functionality in Rel-16 is based on the emulation of LAN features in the 5G core network. FUDGE-5G will go one step further by extending 5GLAN support to unified access under one address “all-Ethernet” domain, opening up the capability to deliver content more efficiently to end users consolidating multiple existing networks into a single converged 5G network.

FUDGE-5G will integrate the required network functions to enable 5GLAN for connecting several access networks (5G, Wi-Fi and fixed Ethernet), providing a unified all-Ethernet address domain where devices are able to interact seamlessly to other devices inside the private network.

2.2.1 5GLAN over 3GPP Access Technologies

The deployment of 5GLAN over 3GPP access technologies requires a set of new network functions and the extension of some NFs with additional functionalities such as UPF.

The UPF will include the 5GLAN network adaptation to integrate mobile devices as part of fixed LAN networks. A new UPF that includes a 5GLAN adaptation layer would be part of the SBA. Thus, the 5GLAN NF would be registered following the SBA architecture to be accessible from the AMF and SMF to setup the end-to-end connection between mobile devices and other fixed devices connected to the LAN.

The 5GLAN UPF will be part of the SBA to be utilised in the FUDGE-5G platform, featuring end-to-end transport between mobile and fixed devices. Of particular importance is the exposure of a 5GLAN NF to application developers, therefore realising the fully virtualised Layer 2 connectivity extending the LAN capabilities to a mobile telecommunication system for the first time.

The 5GLAN UPF will support Ethernet PDU type data. Other NFs such as SMF together with UPF shall also support Allocation and Retention Priority (ARP) proxying as specified in Internet Engineering Task Force (IETF) Request for Comments (RFC) 1027 and/or IPv6 Neighbour Solicitation proxying as specified in IETF RFC 4861 functionality. Moreover, the SMF will interact with the UPF during a PDU session to request the UPF acting as the PDU Session Anchor to proxy ARP/IPv6 Neighbour Solicitation or to forward the ARP/IPv6 Neighbour Solicitation traffic from the UPF to the SMF.

Thus, 5GLAN UPF will include additional packet modifications in order to integrate devices as part of the 5G LAN. The UPF needs to modify the following fields for the (from gNB) received messages: Ethernet Preamble, Start Frame Delimiter (SFD) and Frame Check Sequence (FCS), which are not sent over the 5G network. The UE will remove the Preamble, SFD and FCS from the uplink traffic Ethernet frames, those fields shall be added by the UPF acting as the PDU Session Anchor. The downlink will be modified and the 5GLAN UPF acting as the PDU Session Anchor shall strip the Preamble, SFD and FCS from the Ethernet frame.

In LAN environments, the IP address is not allocated by the SMF to the UE for this PDU session but instead might be allocated by DHCP server located in the fixed LAN. The UPF shall store the MAC addresses, received from the UE, and associate those with the appropriate PDU Session.

The other functionality required for LAN integration into a 5G network infrastructure is the creation and management of private 5G Virtual Networks (5G VN). The 5GLAN Group may be dynamically created by an operator or possibly requested by the AF via service exposure (NEF). The information of 5G VN group is provided by the AF to the NEF and stored in the UDR, by using the NEF service operations information. The NEF might include the Group Management Function (GMF). This GMF will implement the functionality within the NEF for creating, modifying, or removing a 5GLAN Group, according to authorised request from the UE or the AF. GMF in the NEF may store the 5GLAN group information in the UDR. Once

the 5G VN is created, the UE will be accessing the 5G VN with a PDU session that is established for a specific 5G VN. Thus, the 5G LAN functionality would be provided by new features added to different network functions i.e., NEF, UDR, PCF, SMF and UPF that allow the creation, management, and access to 5G Virtual Network.

2.2.2 Relationship to FUDGE-5G

5GLAN feature enables the integration of mobile networks as a part of existing IT infrastructure. Connectivity based on 5GLAN reduces the use of Ethernet cables and provides similar connectivity to autonomous vehicles. For traditional Ethernet communication, a device needs to find out the MAC address of its peer device. The device, according to the destination IP address XXX derived from the IP packet that needs to be delivered, would initiate an enquiry "who has the IP address XXX", and this enquiry is broadcasted to all the devices belonging to the same LAN. The device who has this IP address will respond and provide its MAC address to the requesting device.

For 5GLAN case, it is essential to allow a UE to obtain the identifiers of other UEs in the same private communication of 5G LAN-type service for application communication use. In LAN networks, devices make use of discovery mechanism (e.g. Bonjour or UPNP) to discover other devices online to be used and their characteristics. This discovery mechanism makes use of the multicast capabilities of the network. Therefore, it is important that 5G LAN support discovery mechanisms. The 5G network shall support the routing of non-IP packet (e.g. Ethernet packet) efficiently for private communication between UEs and Control Center (UE). On-demand establishment of a multicast communications within a subset of UEs that are members of the 5G PVN, e.g. Equipment A creates a multicast on demand and B and C joins this multicast to receive A's multicast messages.

The configuration of the UE is performed from a logical Application Function (AF) that configures the UE via the PCF directly or indirectly via the NEF first and the PCF second.

2.3 Time Sensitive Networking

Time Sensitive Networking (TSN) goes back to the concept of "circuits". Instead, they are called "streams", where networks should provide a network-wide precision clock reference to limit network delays to a well-known preferably small value. TSN requires deterministic data delivery with guaranteed delay instead of "best effort" with "connections" highly adaptive and retries. The TSN network should separate non-time-sensitive traffic from time-sensitive traffic to reduce delay and delay variation. This can be done by using network slicing as defined in 5G architecture.

TSN specifications for fixed networks are defined in the IEEE 802.1 Time Sensitive Networking Working Group (TSN WG) [IEE21]. The TSN WG has defined the required

specifications for queuing and forwarding time-sensitive streams. Some of these specifications are the 802.1Qav credit-based shapers, new P802.1Qbu pre-emption, P802.1Qbv time-aware queuing, P802.1Qch cyclic queueing, P802.1Qci input gating, and P802.1CB seamless redundancy [TEE08].

TSN needs to guarantee reliability so registration and reservation of time-sensitive streams is required. Thus, for resource reservation, TSN WG has specified 802.1Qat, which provides – a distributed "stream reservation protocol", extended in new P802.1Qcc to support preemption, scheduling, centralised control, and interaction with higher layer IETF services. The key enabler for TSN is time synchronization, which has been specified in IEEE 802.1AS (based in turn on the IEEE 1588 specification).

Naturally, TSN has been specified for fixed networks, and therefore using TSN over 5G needs an adaptation. 5G has defined a set of network components to fulfil the requirements specified by TSN WG and the associated specifications. 5G networks shall support a restricted set of UEs to communicate privately amongst each other even if these UEs are subscribers to different MNOs.

The 3GPP system shall support on-demand UE to UE private data communication connections with multiple types of data. At least IP and Ethernet should be supported for the TSN devices to natively communicate across the 5G system. The 5G network shall enable MNOs to create a 5G Virtual Networks (VN) for one or more UEs similarly to 5G LAN support. The 5G network shall be able to provide the required QoS (e.g., reliability, latency, and bandwidth) for non-IP packets (e.g., Ethernet frames) in private communications between UEs.

Therefore, the first requirement to be supported by the 5G system is to support accurate time synchronization so the UEs can deploy deterministic communications. Proper and reliable time synchronization is required. To avoid the fact that a later transmission arrives before an older one, the actual latency for each packet delivery should be stable. 5G defines two translators to be integrated in the UPF and the UE to fulfil the 8021.AS requirements for exchanging time synchronization messages. These TSN translator modules are the Device-side TSN translator (DS-TT) and the Network-side TSN translator (NW-TT). These translators at the edge of the 5G system need to support the IEEE 802.1AS and fulfil a series of functions such as Generic Precision Time Protocol or (g)PTP support, timestamping, Best Master Clock Algorithm (BMCA), and rateRatio.

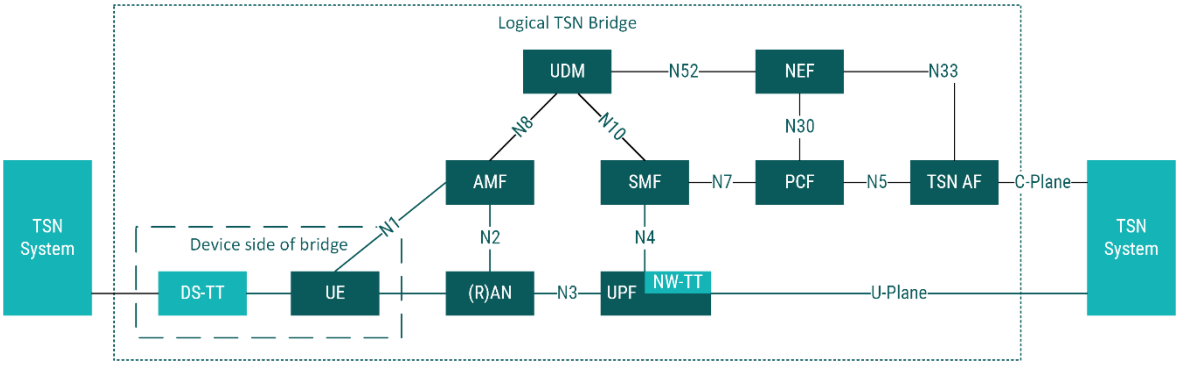

Figure 2.2: Time Sensitive Networking Architecture

Note that the UE, gNB, UPF, and both translators NW-TT and DS- TTs are synchronized with the 5G grandmaster (GM) (i.e., the 5G internal system clock, 5GS).

2.3.1 Precision Time Protocol as part of 5G Time Sensitive Networking

(g)PTP is one of the key enablers of 5G-TSN. It is a protocol that permits the synchronisation of clocks in different devices across a specific network. PTP v1 was defined in 2002 under the name IEEE 1588-2002 [IEE02]. The second version, known as PTP v2, was later defined in IEEE 1588-2008 [IEE08] to be used in a large variety of applications. Timestamping and time synchronization are some of them. Since then, several profiles have been described for some applications and verticals. Note that IEEE 802.1AS is an adaptation of PTP for TSN.

Each application may have different requirements, attribute values or optional features to be achieved using PTP. For each one of them, a set of parameters is set, which constitutes the PTP profile that is used by organizations to set the requirements [MIC17].

From an industrial point of view, one of the most important requirements is time synchronization. It is defined as the time difference between two clocks that are located in different devices or parts of the network. In 5G TSN, there are two types of clocks (TSN and 5GS). The first one is specific for the device and the second clock is generic and used for the entire system. In this case, time synchronicity is applied by updating the child clock (TSN) at the device, when applying the difference between this clock and the (5GS) master. Generally, PTP achieves a clock accuracy in the range of nanoseconds to microseconds, depending on the use case and target value.

PTP also provides timestamping by using message editing, e.g. time marks such as the 5G GM could be inserted as part of an IP datagram. For this purpose, in child-parent systems, PTP is performed in five steps. First, the master sends a synchronization message at a specific moment (t_1). This message is received by the child (t_2), which also knows the time t_1. This is because the master sends a *Follow_Up* message that contains this data. Afterwards, the child sends a new *Delay_Req* message (t_3), which is received by the master and timestamped (t_4).

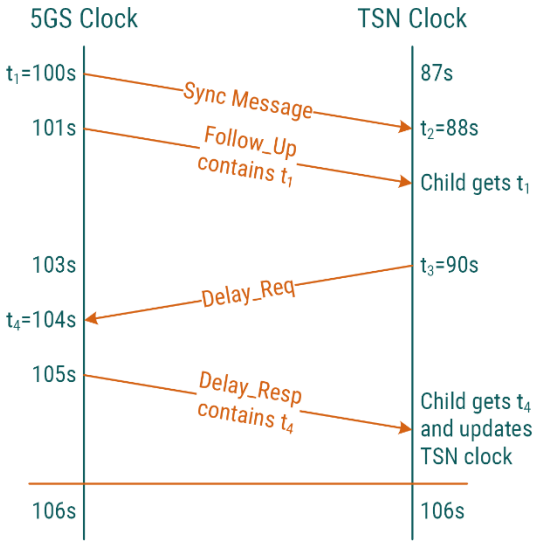

Figure 2.3: Precision Time Protocol: timestamp updating process.

The child clock then needs to know this value (t_4) to perform the final update, since t_1, t_2 and t_3 are already known. Hence, the master sends a last message with the timestamp. Finally, the child calculates the offset value as follows:

$$offset = t_2 - t_1 - \frac{1}{2}((t_4 - t_1) - (t_3 - t_2))$$

Once this offset is obtained, the child updates the clock by adding it to the previous value.

2.3.2 Relationship to FUDGE-5G

TSN requires 5GLAN functionality and utilises Network Slicing. Time sensitive traffic is separated from the best effort traffic and managed using specific Application Function. TSN enables synchronization of 5G network with IEEE 1588 networks that is used in fixed Local Area Networks. 5G technologies enable significantly more flexible implementation of networks and thus provide more flexible factory lay-outs. 5G is also used in mobile autonomous vehicles. Using TSN 5G UEs can be synchronized with the factory infrastructure using one central clock source.

2.4 5G Multicast

Multicast is known as the ability to send information from one node to several nodes without resorting the replication of each data stream per node. In this sense, multicast provides an efficient way to deliver data inside a transport network. 3GPP included multicast/broadcast capabilities since 2006[HAR09], known as Multicast Broadcast Multimedia Service (MBMS). MBMS is a set of extensions over the baseline, point-to-point Core Network to enable point-to-multipoint transmissions oriented towards Digital Terrestrial Television (DTT) services, both at transport and radio layers. MBMS features

have been expanded over past 3G and 4G releases; the most relevant being the inclusion of Multicast Operation on Demand (MooD), Single Cell Point-to-Multipoint (SC-PTM), and LTE-Based 5G Terrestrial Broadcast. Note that the 4G version of MBMS is renamed as enhanced MBMS or eMBMS [CAL15].

From Release 15 onwards, the efforts in standardizing Multicast/Broadcast transmissions can be separated into two different strands: a Broadcaster oriented one, based on LTE and radio modifications to adapt to existing infrastructure (High Power High Tower [BAR20]) and normalized in Release 16; and a Broadband oriented one, based on 5G technologies and oriented towards dense, Low Power Low Tower deployments and currently being standardized in Release 17. This 5G version is known as Multicast/Broadcast Services (MBS).

This section details the second strand mentioned above, and the current standardization efforts to include Multicast functionalities into the 5G System.

2.4.1 IP Multicast in 5G

The work inside 3GPP to standardize Multicast transmissions is divided into three different items:

- TS 23.247: Architectural enhancements for 5G multicast-broadcast services. This document reflects the normative work to include Multicast capabilities into the 5G System, at transport level. TR 26.802: Multicast Architecture Enhancement for 5G Media Streaming. 5G Media Streaming is an integration model between content providers and operators that aims to enable high quality downlink stream services for mobile users. This report focuses on the expectations and issues to interconnect the Media Streaming extensions into the Multicast framework.
- TR NR multicast and broadcast services. Changes to the NG-RAN necessary to accommodate the multicast transmission modes. Note that physical layer changes to NR are not expected, in order to ease the adoption of multicast among chip manufacturers.

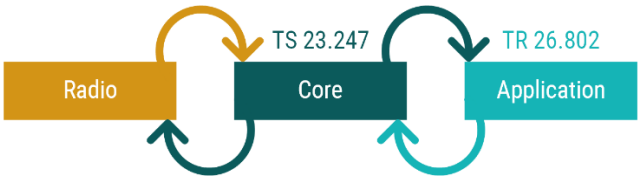

Figure 2.4: Relationship between 3GPP documents and multicast architecture

5G Multicast inherits several legacy concepts and identifiers from 4G eMBMS, such as MBS sessions, communication services, service area and Temporary Mobile Group Identity (TMGI); and adds new ones like the traffic delivery models, split into Individual and Shared

MBS traffic delivery, and the 5G QoS models (5QI). Figure 5 shows the relationship between the different terms.

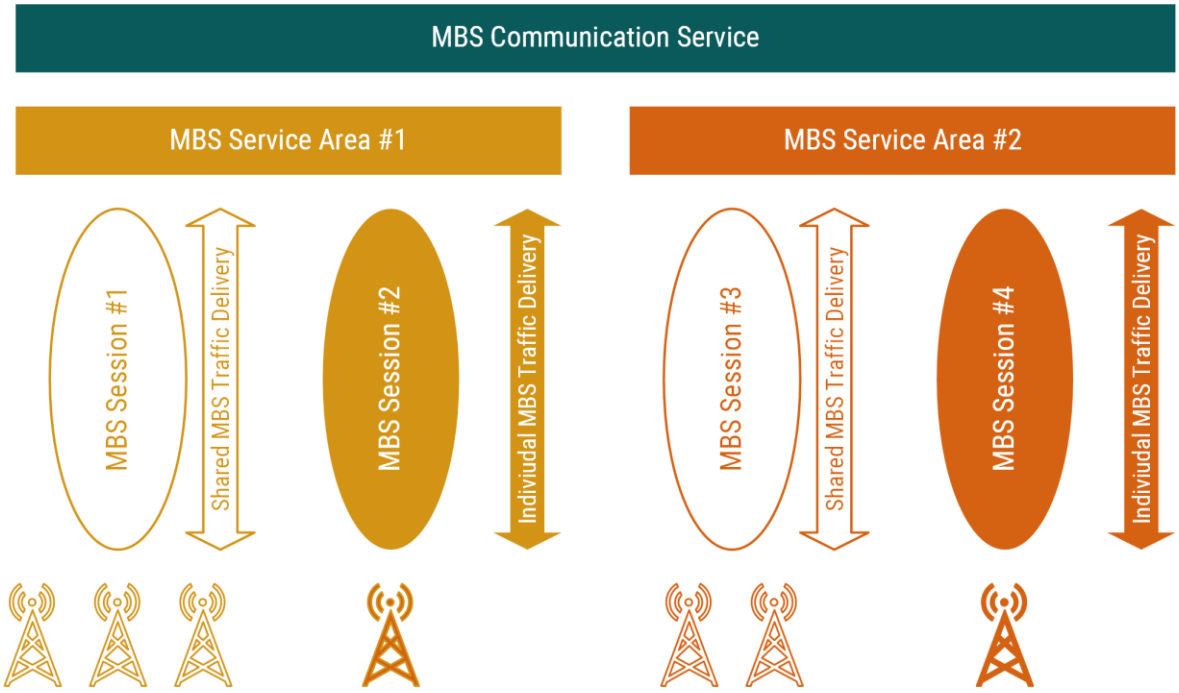

Figure 2.5: Example MBS concepts and their relationship.

5G Multicast recovers the multicast communication capabilities, introduced in the 3G version (MBMS) but abandoned in the 4G version (eMBMS). This allows for point-to-multipoint communication between a specific set of UEs, who must join the multicast communication. On the other hand, broadcast communications are permanently transmitted under a cell and any device in coverage can tune and receive the broadcast content. Multicast transmissions are transparently delivered to the devices, this means that the UEs reuse the unicast procedures to get the content and are not necessarily aware that the content is being multicast.

To support Multicast/Broadcast communications, new additions and modifications of existing Network Functions are added into the 5G Core architecture. Inside the transport layer, MB-UPF and MB-SMF replace the UPF and SMF respectively. At the service layer, optional network functions MBSF and MBSTF provide the necessary application-level functionality to monitor, manage and anchor the MBS Communication Service(s). Figure 6 describes the reference architecture, with dashed boxes being optional:

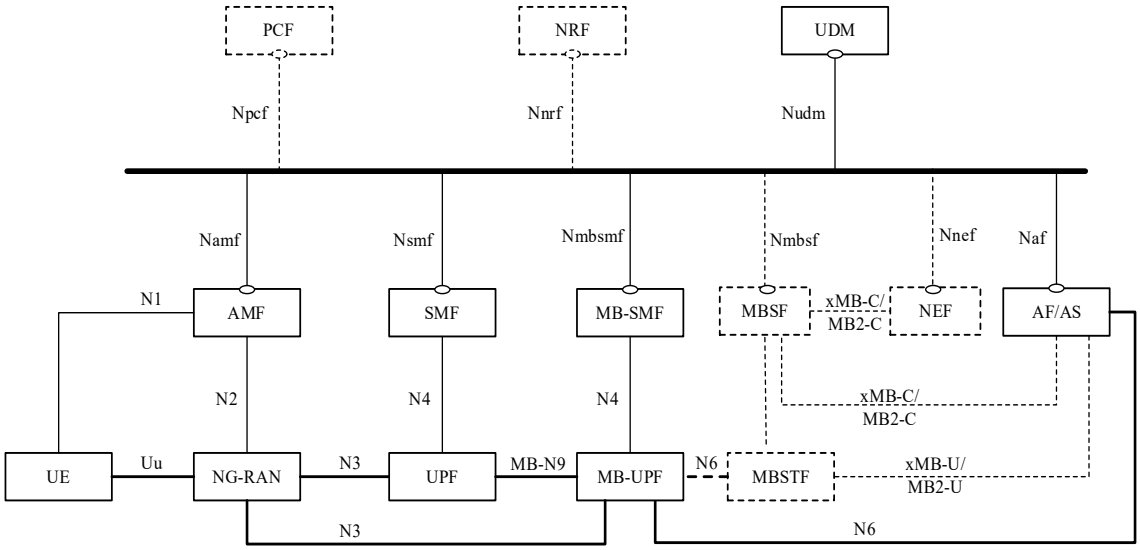

Figure 2.6: Reference architecture for MBS in Release 17 [3GP21]

The existing reference points of N1, N2, N10, N11, N16, and N29 are enhanced to support 5G MBS. The existing reference points of N7, N18, N29 and N33 are enhanced to support 5G MBS depends on deployment. The N4 reference point between NEF (MBSF) and MBSTF is enhanced to support 5G MBS service level management required by AF/Application Server.

2.4.2 Opportunistic Multicast

Opportunistic multicast is defined by the capability to reintroduce multicast packet delivery behaviour for HTTP communication. HTTP, as the dominant application protocol for web services, has pure unicast characteristics for both the request and response between endpoints. The benefit of delivering payload intense HTTP responses, e.g. video or software updates, in a multicast fashion through the network would significantly free up networking resources. The technology that can provide such capabilities is entitled Name-based Routing (NbR) [TRO19] which is based on Information-centric Networking (ICN) principles where a name, e.g. host field within a HTTP request, is being used to identify the endpoint that can serve the content of the packet. The forwarding of packets between SPs utilises Bloomfilters with each link in the network being represented as a bit and each SP injects packets with forwarding identifiers that have all links the packet is supposed to traverse set to 1 in the bitmask that is read by each switch on the way. This operating is referred to as arbitrary bitmask matching in comparison to longest prefix matching (aka IP). By apply this path-based forwarding scheme, the routing of any packet in an NbR system is fully decoupled from the identifier of the endpoint, which for IP routing is not the case.

NbR utilises a publish-subscribe model to achieve the matching of HTTP client and servers based on said host field in HTTP requests. Furthermore, NbR operates transparently to standard IP endpoints by acting in a proxy-fashion at the ingress and egress points of the

network with the ability to even proxy the IP and ICN world directly on the client or server. Theses proxies are called Service Proxies (SPs) and have no hardware dependency making it an entirely softwarised technology. The NbR system architecture is Figure 2.7 and illustrates the IP endpoints *Client* and *Server* as well as the ingress and egress Service Proxies SP_C and SP_S. The actual publish-subscribe functionality and logic takes place inside the Path Computation Element (PCE) which also acts as a northbound SDN controller application to configure the SDN switch port rules.

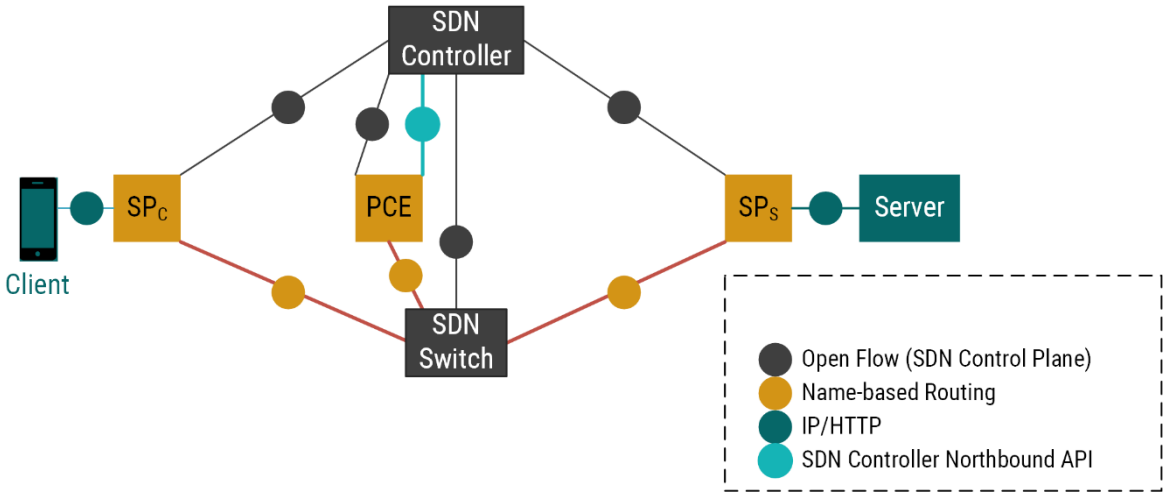

Figure 2.7: System architecture of Name-based Routing with SDN integration

Based on the publish-subscribe ability, NbR can match more than one subscriber to a publisher. This allows to have a particular packet to be sent to more than one endpoint and the ability to reintroduce multicast behaviour due to the fact that routing is fully decoupled from the publish-subscribe behaviour. NbR utilises a path-based forwarding technology based on bitfields allowing to pre-determine how the packet traverses the links in a network with the ability to have a packet being sent out on multiple ports on any intermediate switch in the topology.

When a Service Proxy (SP_S) receives a HTTP response it has the ability to perform a publish-subscribe action across all clients that await the response. Thus, if more than one client awaits the HTTP response the packet(s) comprising the response can be sent in a multicast fashion to all SP_Cs where IP stack-compliant packets are issued to the client(s) served by the SP_C.

If the HTTP session is using Transport Layer Security (TLS) [RES18], SP_Cs can act as TLS endpoints or proxies. After the TCP session has been established by the UE with the server, which is transparently intercepted by the SP_C, the UE initiates a TLS handshake aiming to establish a secure connection. When the SP_C receives the ClientHello message from the client it has two choices: act as a TLS endpoint or act as a TLS proxy. If the root certificate was made available to NbR by the application provider, the SP_C acts as a TLS endpoint (full TLS interception) and can offer all service routing capabilities for HTTP. If not, the SPC acts

as a TLS proxy (not intercepting TLS – only proxying packets) and offers only a subset of the service routing capabilities for HTTP traffic such as shortest path routing.

2.4.3 Relationship to FUDGE-5G

Multicast/Broadcast provides scalability in the transport and radio network. Big amount of shared data can be delivered without incurring into high network resources; and thanks to the System Slicing paradigm, the specific instantiations of the Multicast-based Network Functions required to enable this functionality do not interfere with other services coexisting in the same infrastructure. Not only limited to multimedia delivery, multicast can be used inside a factory floor to mass deliver a firmware update to thousands of sensors, or as an evacuation alarm system to devices under a NPN.

2.5 End-to-End Slicing in NPNs

A network slice is a self-contained end-to-end (E2E) logical network, ensuring resource isolation, and including all necessary functions tailored to a given application or service. Network slicing makes it possible to offer radio, networking and compute resources to vertical operators without the need for them to have a dedicated physical network infrastructure. For this reason, slicing enables service differentiation by customizing network operation to meet the requirements of customers. It facilitates as well the co-existence of multiple dedicated, mutually isolated E2E slices.

The concept of network slicing has evolved in the last decade; from a simple network overlay used in federated testbed research into a fundamental feature of 5G systems, covering access, core, and cloud networks. This evolution lies its foundations on the emergence of technological enablers such as softwarization, a generic term for virtualisation of the network functions, as well as software-defined programmability of the network functions and infrastructure resources.

Initially, slicing has been introduced within the Next Generation Mobile Network (NGMN) alliance. According to their definition, a 5G slice is a **collection of network functions and specific RAT settings combined together for a specific use case** [NGM15]. From a different perspective, 3GPP defines a slice rather as a **logical network that provides specific network capabilities and network characteristics** [3GP16]. The International Telecommunication Union (ITU) also considers network slicing as an enabler for Logically Isolated Network Partitions (LINP) with a slice being considered as a **unit of programmable resources, such as network, computation and storage** [ITU12]. Finally, GSMA defines network slicing as the **capability to run multiple logical networks as virtually independent business operations on a common physical infrastructure in an efficient and economical way** [GSM17].

From the business perspective, a slice maps to a collection of network resources, functions, and assets managed coherently to fulfil a specific business service. Business-oriented

services being most of the time E2E, different per-domain slices exist often stitched together to create the required E2E services. As an example, an E2E slice can be composed by two domain-specific slices, one internal to the network operator and the other external. The internal slice is completely opaque to the customer, which is provided only with an SLA, and represents the network partitions inside an operator's network, where the operator retains full authority. On the other hand, the external slice can appear to the customer as dedicated networks/clouds/data-centres where the control is shared between the operator and the customer. Inherent to its definition and business objective, any slice faces a continuous tension between the user-centric expected service level agreements (SLAs) and the infrastructure-level network available resources and capabilities.

2.5.1 Slicing Architecture

According to [BAR20a], the basic principles required for enabling network slicing and its related operation on 5G networks include:

- **Automation of network operation,** to manage the lifecycle and adaptation of slices and its components.
- **High-Reliability, Scalability and Isolation**, to ensure performance guarantees and security for each customer.
- **Programmability**, to simplify provisioning and management towards on-demand service customization.
- **Hierarchical abstraction**, to facilitate service provision via logical abstraction of resources and service modelling.
- **Slice Customization**, both at the Control- and User-plane, to provide service-tailored functions and fine-grained forwarding control.
- **Network resource elasticity**, to scale allocated resources to the need of the service.

An E2E slicing framework takes advantage of these principles to interact with multiple domains at once, each domain providing specific types of resources and capabilities (e.g., network, compute, storage). The slicing framework has to negotiate first, and then enforce consistent E2E policies in any of the local domains.

Exemplary domains include RAN, Edge, Core, as well as Cloud. Within a single domain, orchestration is performed on a single type of resource (e.g. NFV [DRA17], RAN [COR19], [KAT16]) through a domain manager (e.g., RAN controller, SDN controller, transport network controller, WAN controller). Closed-loop procedures for resource fulfilment and assurance, or network intelligence make up the building blocks within each one of the domain managers, together with monitoring primitives and APIs for the interaction with the slicing framework. Domain-specific controllers can be instructed to execute policies and rules both at the resources and at functional level. Those domain controllers are, in principle, capable of operating autonomously (i.e., without being programmed by the E2E

framework, and by relying only on their local control loop procedures) to guarantee a minimal, yet degraded, service level in case of network partitioning.

In short, to enforce E2E policies, the slicing framework has to interact with the programming interfaces exposed by each of the domain managers that are then responsible of managing resources locally, handling the life cycle of deployed services, and mapping the E2E slices to the assigned services and resources.

Such conceptual scheme translates into a high-level architecture where an OSS/BSS is a logically centralised entity driving the behaviour of the entire system, by interacting with the (locally deployed) domain managers. The capability of handling multiple administrative domains, each one exposing different resources, ensures the coherence of the E2E slice. Business requirements defined by a customer are translated into Network Slice Description (NSD) as soon as a slice request is made. The NSD is then employed for deployment. Domain managers perform resource allocation to slices based on their demands and priorities. A multi-domain orchestrator handles the life-cycle management of E2E slices across multiple administrative domains while domain-specific managers build slices of the network, compute, and storage resources.

2.5.2 Existing Projects

The ETSI MANO provides a normative framework defining life-cycle management capabilities and configuration of functions as required in a slicing framework [ETS17]. On top of this reference architecture, multiple projects have provided reference implementations. Among the existing approaches stand out the ETSI OSM, ONAP, and OPNFV projects creates a reference NFV platform to accelerate the transformation of enterprise and service provider networks. Related MANO frameworks and 5G architectures for 5G network slicing that considers the management and orchestration of both virtualized and non-virtualized functions but are related to NFV and not radio.

2.5.3 Relationship to FUDGE-5G

With Network Slicing being one of the key innovations in 5G mobile networking, it is of great importance to have covered it in the technology background section to demonstrate the various sub-systems a 5G system is composed of (RAN, 5GC, DN). When realising that a user plane traverses all listed 5G sub-systems, it becomes clear that orchestrating all compute, networking, frequency and storage resources in an end-to-end fashion across a multi-tier environment is challenge that requires a flexible and modern system at its foundation.

2.6 Operations, Administration and Management of NPNs

This section provides details on how current 5GC and vertical applications are provisioned, administrated and managed. The focus here lies on providing an overview about technologies used to achieve the various OAM tasks and demonstrating the amount of information that must be configured annually in a non-programmable manner.

2.6.1 OAM for 5G Cores and Vertical Applications

This section provides details on how current 5GC are operated, administered and managed.

2.6.1.1 Fraunhofer FOKUS Open5GCore

Open5GCore acts as the 5G testbed and is the implementation of 5G core network. This platform follows a modular architecture, where all the 5G core components are implemented as modules and they interact with each other using bindings. The 5GC supports multi-threading, so that parallel execution of core functionalities can be supported.

Prometheus is integrated with Open5GCore so provide monitoring memory statics and similarly other node exporter can be used to get CPU and other relevant metrics to the Prometheus server. Monitoring can also be done based on logs from the individual components and the commands available via command-line interface of the core.

The Open5GCore can be provisioned with various type of virtualization/bare metal deployment technologies e.g. Kubernetes, OpenStack, KVM, LXC, VMware, Docker, etc. The deployments can be manual or automatic. Due to minimalist operational requirements, O5GC can be swiftly integrated with most of the orchestrators available. It is tested with popular orchestrators like OSM, Sonata and OpenBaton. Although it can be easily integrated, each orchestrator has its own requirements hence no separate packages are maintained for orchestrator integrations. Each technology has its own way of provisioning the instances e.g. for Kubernetes we use helm charts, whereas for OpenStack we use OpenStack cli. Based on the type of technology used, the way of provisioning differs.

Open5GCore is configurable via JSON configuration file. Depending on the provision of the configuration files for the components, the core can start the respective services. Each component can be configured via environment variables, component command-line, and rendered config file. The configuration files support environment variables, hence during provisioning, the configuration parameters can be exported as environment variables or the complete configuration file can be rendered using the JSON configuration templates. The values which are configurable by default are IP addresses, PLMN. But more parameters can be made configurable via env variables.

The core is mainly used in the development/testbed environment where in some cases it is essential to capture specific traffic. For this purpose, Open5GCore is designed to handle different type of traffic on different interfaces. Each NF has multiple interfaces based on its functionalities and requirements. There is a dedicated network for management as well and every component has an interface on this management network.

2.6.1.2 Cumucore

Cumucore 5GC has five interfaces, interface N2 to connect the 5GC to the Radio Access Network (RAN), interface N6 to connect the Cumucore 5GC with Packet Data Network (PDN), interface N6 to connect local data processing (MEC), interface for management purposes (e.g., ssh connectivity) and interface to request dataflows inside network slice. Configurable interfaces can have an IP address from the DHCP server or the IP address can be assigned manually. The default route should be through this interface. NAT and IP forwarding is enabled.

There is a minimum computing hardware requirement to run Cumucore 5GC. Computing environment can be cloud and on premises servers. The user is provided with an image that installs both virtual server and the Cumucore SW. After the installation, the user needs to configure settings via GUI and Ansible playbook.

2.6.1.3 Athonet

Athonet's 5G core solution includes a Management System which provides means for performance monitoring, control and user plane functions configuration, networking configuration, license management and SIM provisioning through a Graphical User Interface (GUI) accessible via a browser. The same configuration can also be done through a set of Application Programming Interfaces (APIs).

Moreover, Athonet's solution has a pre-installed instance of the third-party product Prometheus. The Prometheus instance provides statistics on the CPU, memory, disk usage, I/O, system processes, as well as statistics on the usage of the network functions. The metrics can be accessed by any tool that is able to fetch Prometheus data. Grafana is the recommended tool. A Prometheus data source can be added to Grafana and requires the use of a Bearer Authentication token obtained when logging in to the system.

Alarm lists and system logs can be viewed through a REST API as well.

2.6.1.4 One2many

The commercial CBCF of one2many is a monolithic entity and for the FUDG-5G project a proof of concept is being developed for a CBCF that consists of a number of micro-services. At the present moment there is not yet any OAM concept.

OAM is, amongst others, required for provisioning of the CBCF. The Alert Originator sends a message to the CBCF which contains the text of the message that shall be delivered to citizens in the target area and also one or more polygons, circles and geo-codes that define the target area. The CBCF determines which cells cover the target area. Cells and their coverage area cannot be obtained from the 5G core network and have therefore to be provisioned via OAM. The cell coverage information is provided by the operator's cell planning system. For each cell, the CBCF needs to be provisioned with the Cell ID, its coverage area, the Tracking Area that is configured for that cell and which AMF Set serves that Tracking Area. However, the Tracking Areas that are served by an AMF Set can also be discovered via the Nnrf_NFDiscovery service provided by the NRF.

The CBCF sends a message to each AMF Set that serves the Tracking Areas that are configured for the cells that cover the target area. If SCPs are deployed, the CBCF uses an FQDN for the AMF Set and the SCP delivers the message to the appropriate AMF instance in that set. If no SCP is deployed, the CBCF selects the appropriate AMF instance.

2.6.2 OAM of Vertical Applications

The deployment and provisioning of an AF is currently a semi-automated process, as the software itself can be deployed automatically as a VNF (usually as part of another application or platform), but configurations that depend on the 5GC or the network must be performed manually. While provisioning the AF, the first parameter to be configured manually is the address of the NRF. With the NRF, the AF can register itself and find out about other 5GC components that it may access depending on the agreement and trust set between the AF customer and the network operator.

OneSource is currently using its Mobitrust Situational Awareness platform as an AF, with the goal set to request specific QoS policies for its traffic flows. This process can operate in two different modes, depending on the level of trust with the network operator: direct access to PCF, or exposure of PCF interfaces through NEF. In either case, the process starts with a query to NRF for the discovery of the NF to be contacted. If the target NF is a PCF, 3GPP considers the possibility of using a BSF to assist the NRF to answer with the correct PCF for the slice, for instance. However, to perform the NRF requests, the AF is only aware of information that can be previously obtained from other NFs or manually provided, which means that some details will require human configuration or interactions via non-standard interfaces from the UEs that belong to the same entity as the AF. One of those details is the IP address of the UE, which is the key parameter that OneSource is currently using to identify the UE where it aims to have specific QoS policies for its applicational flows.

The aforementioned QoS policies to be requested from the 5GC are based on a set of requirements and use cases related with the end goals of the Mobitrust platform, which include differentiated priority for traffic related with alerts (e.g., man-down, hazardous environment, etc.), system messages and mission critical sensors (e.g., ECG, RR),

differentiated priority for some users over others (e.g., team leaders in field operations, or a team member selected by the Command and Control Centre) or minimum bandwidth required for some applications (e.g. video to be processed with AI). Below, in Figure 2.8 an example of a request payload is given. This request sets priority for data sensors, followed by control messages from the platform and only then for video. Additionally, a minimum bandwidth is requested for each of the services.


```
{
  "ascReqData": {
    "notifUri": "http://10.0.0.10:8080/npcf-policyauthorization/v1/terminate",
    "ueIpv4": "198.51.100.1",
    "suppFeat": "0",
    "ipDomain": "onesource.pt",
    "afAppId": "mobitrust",
    "medComponents": {
      "1": {
        "medCompN": 10,
        "medType": "DATA",
        "mirBwUl": "1 Kbps",
        "resPrio": "PRIO_1"
      },
      "2": {
        "medCompN": 20,
        "medType": "CONTROL",
        "mirBwUl": "100 Kbps",
        "resPrio": "PRIO_2"
      },
      "3": {
        "medCompN": 30,
        "medType": "VIDEO",
        "mirBwUl": "1000 Kbps",
        "resPrio": "PRIO_3"
      }
    },
    "evSubsc": {
      "notifUri": "http://10.0.0.10:8080/npcf-policyauthorization/v1/notify",
      "events": [
        {
          "event": "FAILED_RESOURCES_ALLOCATION",
          "notifMethod": "ONE_TIME"
        }
      ]
    }
  }
}
```


Figure 2.8: Mobitrust AF Example Request Payload

With such procedures for provisioning and operation, and with very low standardization from 3GPP on the AF side in general, several challenges arise with the operation of an AF. These challenges include the inability to obtain some network information (e.g., DNN) programmatically, which means the process is not fully automated and becomes burdensome. Moreover, there is no base framework for agreements with operators nor a single standardized procedure for obtaining information that is not readily available from the 5GC APIs, leading to many non-standardized interactions that raise OPEX, increase complexity and prevent further adoption from stakeholders.

Up till now there is a gap between the 5G world and the vertical applications. The reasons for that are numerous but the more evident are that the verticals applications are treated as cloud apps. The typical cloud model is insufficient for verticals as these verticals exhibit certain peculiarities (i.e industry apps or automotive related). Another reason is that network and telco operators are out of the whole picture. This occurs because network operators are considered as plain connectivity providers between different sites and the cloud applications.

Considering the above and having as main target to lower the entry barrier for the technology innovators and to greatly facilitate them in bringing their vertical applications we introduced an application driven orchestrator. The goal of the application driven orchestrator is to offer a unified framework for the quick adoption of verticals application that are applicable in real 5G industrial environments. The enabling features for this are the mechanism for interfacing the vertical application requests as use case specific application graphs and the application graph repository framework that enables the quick update and redesign of a vertical application use cases or even the creation of vertical application as combination of others. The latter allows the implementation of a quick and automated process in the deployment of new services. This process in addition to the extended application driven orchestration framework constitute specific and valuable outcomes from the project that will be directed towards standards development.

2.6.3 OAM of Multi NPNs

The multi NPN is essentially a private network deployed and capable of connecting to other private networks. In case of multi NPNs, the provisioning and configuration for a private network remains same as explained in 2.7.1.1 for Open5GCore. In case of interconnected NPNs, the SCP need the information of peer SCP's, which can be configured via configuration file of SCP or command line on SCP component. The core is considering the underlying network is taking care of security between two deployments as the core itself does not provide any security on communication layer.
The administration of the NPN deployment is not centralized hence each deployment is individually configured, maintained and monitored.

2.6.4 Relationship to FUDGE-5G

This section provided a brief insight into OAM procedures for 5GCs and vertical application deployments. It can be observed that the landscape of approaches, technologies and level of automations is rather scattered and the SBA vision of cloud native deployments that utilise programmable orchestrators to achieve this goal. As FUDGE-5G aims at providing an evolved SBA platform offering cloud native orchestration capabilities, the overview provided in this section is of great importance and provides the argumentation of the necessity for a unified cloud native orchestration enabling multi-vendor NPN deployments.

3 System Architecture

This section presents the system architecture of FUDGE-5G focussing on key innovative aspects of the project. The section is structed around the high-level architectural blueprint described in the next section and dives into each blueprint component block individually.

3.1 High Level Architecture

The high-level FUDGE-5G architecture illustrated in Figure 3.1 depicts three layers: The **infrastructure layer** is concerned with components and technologies that are assumed to be available in an operator's infrastructure and exposed through standardised and open APIs. Within the infrastructure layer FUDGE-5G assumes a unified access domain, in the likes of 802.3 as the common denominator as the frame format. If a switching fabric is available in the operator's network, FUDGE-5G assumes it to be fully programmable via Software-defined Networking (SDN) procedures, e.g. OpenFlow. As the FUDGE-5G platform is fully softwarised, all its components can be virtualised and provisioned as Virtual Network Functions (VNFs). Thus, the FUDGE-5G platform can either be provisioned on native Commercial off the Shelf (COTS) systems without any virtualisation or via an infrastructure orchestrator following the ETSI MANO reference model.

The FUDGE-5G **platform layer** is composed of the three functional blocks routing, service orchestration and monitoring. The routing block comprises the two functions service routing and resource scheduling. While service routing is concerned about the ability to perform fast and adaptive service routing among Cloud Native Network Functions (CNFs), resource scheduling is performing decisions on which CNF service instance to be chosen from a pool of one or more available instances. These decisions can implement various optimization criteria in order to meet specific quality of service aspects, such as distributing load equally over a set of CNF instances, limiting the delay of service invocations or similar. The service orchestration block provides location-aware cloud native orchestration for CNFs and an additional vertical application orchestrator for – as the name implies – vertical applications, also utilising the SFV orchestrator. The third block inside the platform layer is concerned with telemetry which is composed of a cross layer and vertical application monitoring as well as an analytic functionality.

The **service layer** is divided into the two areas 5GC and vertical applications. While the 5GC lists innovations of the FUDGE-5G project around 5G Opportunistic Multicast (OMC), 5G Time Sensitive Networking (TSN), 5G Local Area Network (LAN) and interconnected NPNs, the vertical applications range cover four of the five use cases realised by FUDGE-5G to demonstrate the innovations, i.e. media, industry 4.0, public protection and disaster relief (PPDR), and virtual office.

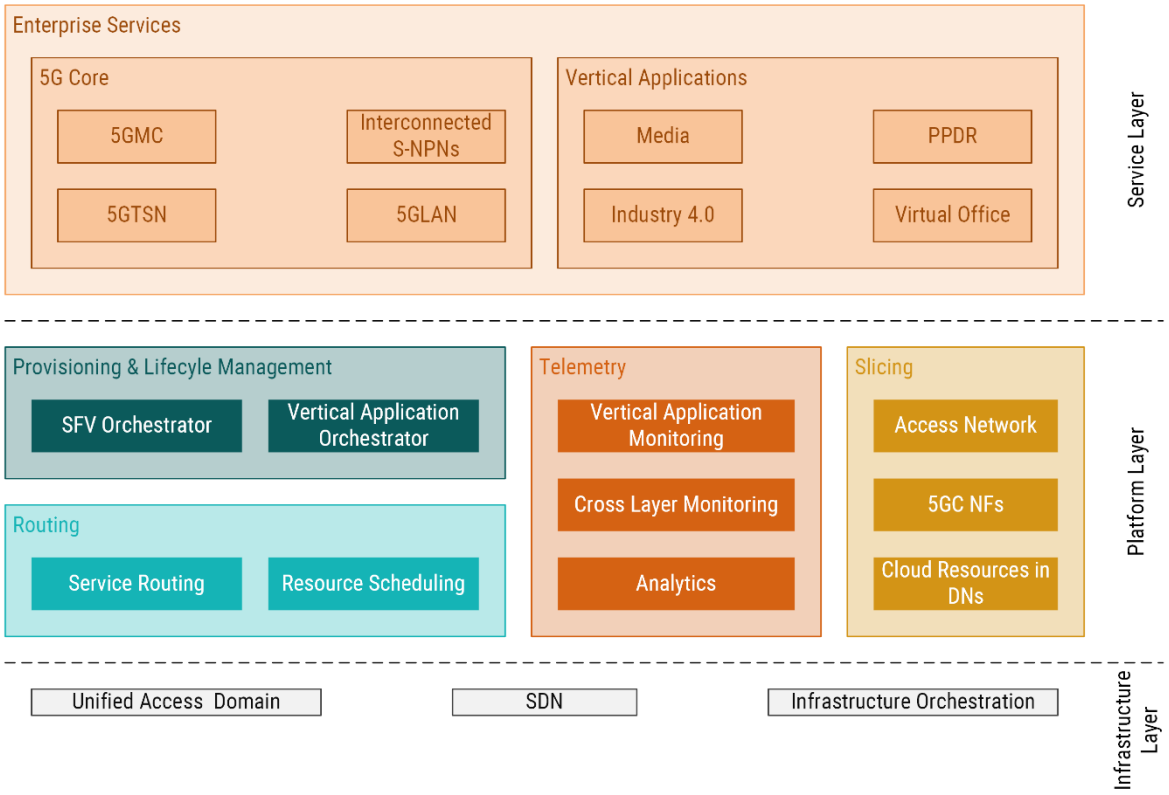

Figure 3.1: FUDGE-5G high-level system architecture

As stated in 3GPP's architecture specification [3GP19b], the SCP is an addressable component that is part of the 5GC and enables *indirect communication* among SBI-enabled NF services. In case when no SCP is in place, the standard describes that as a *direct communication*. This communication also offers routing capabilities. However, when closely comparing this definition with how public cloud offerings and how the internet is being used, two clear distinctions can be observed:

- In 3GPP, the SCP – as the component that routes traffic between consumer and producers – is directly addressable by using an FQDN or IP address. This stands in stark contrast to internet routing semantics where the routing layer is never addressed directly. Instead, the service is addressed by means of an IP address (through a DNS that resolved an FQDN where applicable).
- In an architecture figure illustrating the components of a cloud native service, e.g. 5G Core, the routing component belongs to the underlying platform which offers the orchestration (provisioning and lifecycle management) as well as monitoring and routing capabilities. In line with the cloud native proposition, this allows the realisation of platforms in a service-agnostic fashion.

Based on the points above, Figure 3.2 illustrates the resulting change to 3GPP's architectural illustration where the SCP is not presented as a 5GC Network Function and includes the N4 to Nupf transitioning. Furthermore, all SBI-enabled NFs will require to

accept the existence of an SCP with an addition to the currently three SCP deployment options with a fourth one that does not offer any advanced proxy functionality and acts as an interconnected network of sub-systems, similar to the internet. Also, following the routing semantics in clouds and the internet, FUDGE-5G proposes to remove the ability to address an SCP directly (aka the routing layer). Thus, the differentiation of *direct* and *indirect* communication becomes an obsolete concept that does not require to be upheld.

While the role and proposition of the SCP in FUDGE-5G is a direct conclusion of the service routing work in FUDGE-5G, the Nupf transitioning has its roots in the cloud native orchestration of Enterprise Services such as 5GCs. While an Nupf SBI allows a standardised reporting of monitoring information about the UPF, its importance becomes even more critical when focussing on the provisioning of 5GCs. The FUDGE-5G platform comes with its very own orchestration (i.e. provisioning and lifecycle management) technology targeting 5GCs where the bootstrapping of N4-based UPFs would become a hurdle to overcome. The required UPF and SMF association through manual UPF registration at the NRF or directly in SMFs significantly conflicts with the objective of cloud native orchestration. Cloud native represents the ability to provisioning a service (ideally represented by multiple microservices) across a range of compute hosts and in a programmable manner where an orchestrator offers the provisioning of 5GC NFs as a service. Thus, the ability to separate the orchestration task from application layer is critical for cloud native and service logic such as the association of SMFs and UFPs is a service-layer (5GC) problem and not a platform (orchestration) one. Once the UPF is addressable through a name (FQDN), the FUDGE-5G platform would allow the provisioning and routing among all instances through its orchestrator and routing platform components.

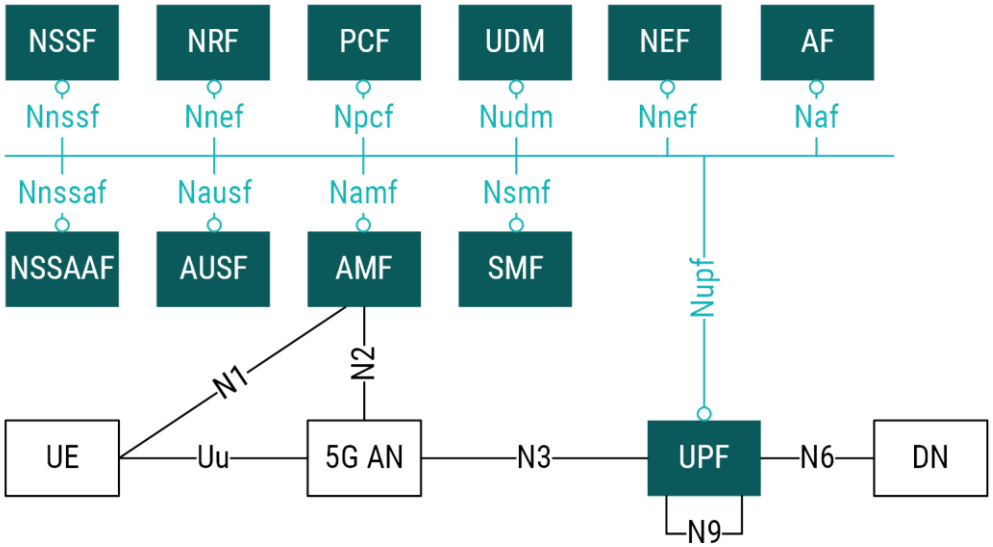

Figure 3.2: Amended (B)5G 3GPP System Architecture

3.2 Routing

The Routing component in our high-level system architecture in Figure 3.1 implements solutions for the following problems:

1. ***Identifier mapping*** is concerned with the mapping of a service identifier, such as a URL for an HTTP-level service, to a routable identifier, i.e., an identifier that can be used directly in the decision to forward the request to a suitable (CNF) instance.
2. ***Flow affinity*** enforces the bundling of one or more packets into a service-level semantic, e.g., an HTTP request (or response). This ensures that packets for the same request (or to those belonging to a flow of requests) are sent to the same CNF instance.
3. ***Rendezvous*** determines the set of possible CNF instance choices, based on the service identifier mapped in Item 1.
4. ***Resource scheduling*** determines the most appropriate choice of CNF instance in the presence of possibly more than one such choice; the capabilities will be discussed separately in Section 3.2.3.
5. ***Path calculation*** takes the output from Item 4 to determine the suitable path from the network ingress to the chosen CNF instance in the form of a path identifier.
6. ***Packet handling*** finally ensures a suitable packet encoding that includes all relevant information, such as the service identifier of Item 1, the path identifier of Item 5, flow affinity information of *i*tem 2 as well as transport network specific information such as maximum transfer unit size to inject a suitable packet into the network for delivery to the chosen CNF instance.

While Item 4 above is realised in the *resource scheduling* sub-component, the solutions to all other aspects form the *service routing* sub-component. Those sub-components are discussed in more detail in the following.

3.2.1 Service Routing on the Control Plane

For the routing on the control plane, FUDGE-5G offers an SCP that is composed of a resource scheduler and service routing for HTTP traffic, as illustrated in the high-level architecture figure. While the resource scheduling component aims at determining the 5GC instance affinities, the service routing component offers the actual routing of HTTP packets in a most transparent and dynamic manner.

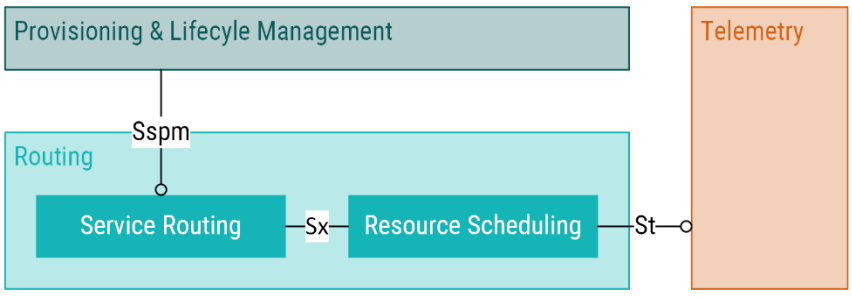

Figure 3.3: Service routing interfaces

The **Sspm** interface allows a – to the service routing external – component to register and unregister IP service endpoints to/from the routing layer. The Service Routing component requires the IP and of the endpoint as well as its FQDN under which is should be reachable by consumers to complete a registration. Full details of this interface are provided in Deliverable 2.1 [FUD21].

The **St** interface allows the resource scheduler to retrieve required monitoring information from the telemetry component in order to determine which consumer and producer instances should communicate with each other ensuring the resources available are sufficient.

Consequently, the **Sx** interface allows the resource scheduler to communicate its decisions to the service routing component. The details of this interface will be provided in the final architecture deliverable D1.3.

3.2.1.1 Intra and Inter Public Land Mobile Network Routing via the Service Communication Proxy

As outlined in Section 2.1, the SCP is an integral part of the 5G system architecture and was specified in Release 16. As FUDGE-5G applies all its innovations in an NPN setting, this section briefly outlines the two modes foreseen to cover all use cases.

Following the definition of an SCP (Release 16 and 17), it enables indirect communication for SBI-enabled NFs within a PLMN. As some FUDGE-5G use cases pose the question of interconnecting numerous S-NPN deployments where the Public Network Integrated-Non-Public Networks (PNI-NPN) is not an option, this section provides an overview of how to achieve this.

3.2.1.1.1 SCP-based Intra Public Land Mobile Network Routing

This scenario is in-line with what Release 17 foresees for the usage of the SCP, i.e. an indirect communication capability for NFs within a PLMN. Figure 3.4 illustrates two S-NPNs that are represented by different PLMN IDs, i.e. PLMN 1 and PLMN 2. The SCP is solely used for the indirect communication for SBI-enabled 5GC NFs within a PLMN without any connection to outside the PLMN.

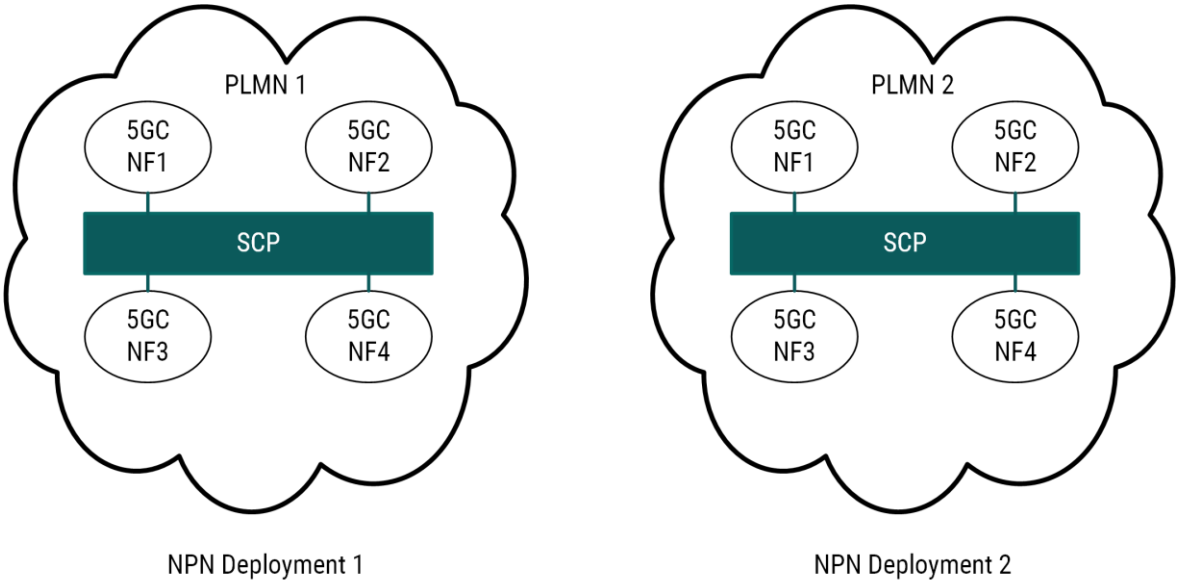

Figure 3.4: SCP-based intra domain deployment

3.2.1.1.2 SCP-based Inter Public Land Mobile Network Routing

For interconnecting S-NPNs and allow the roaming of users across S-NPNs a new component is introduced to enable inter PLMN rouing, i.e. the Session Border Controller (SBC). The SBC's functionality can be seen as a mix of SEPP and SCP. In the 3GPP Release 17 specifications, an SCP is referred as a control plane proxy and solely used to forward messages between the NFs within a single domain. But in order to support roaming, the SBC is being extended by the defined features of an SCP. The SBC uses SCP features to route messages between multiple domains. Also, integrating the features of SEPP to accept messages only from authorized domains.

SCP is implemented as a Network Function (NF) and is responsible for forwarding and routing messages to destination NF in the destination network identified by Public Land Mobile Network (PLMN). In Section 2.2 of this document, there are some deployment options provided for SCP, and for the use case, Interconnected NPNs, the SCP will be deployed as independent deployment units and each domain will have an SCP deployed.

For Intra PLMN communications, direct communication is used for the message forwarding. The NF consumer asks for the profile of the NF producer to the local NRF. NRF uses the discovery parameters from the request and replies with the corresponding NF profile. The NF consumers use the details from the NF profile and forwards the message to the NF producer.

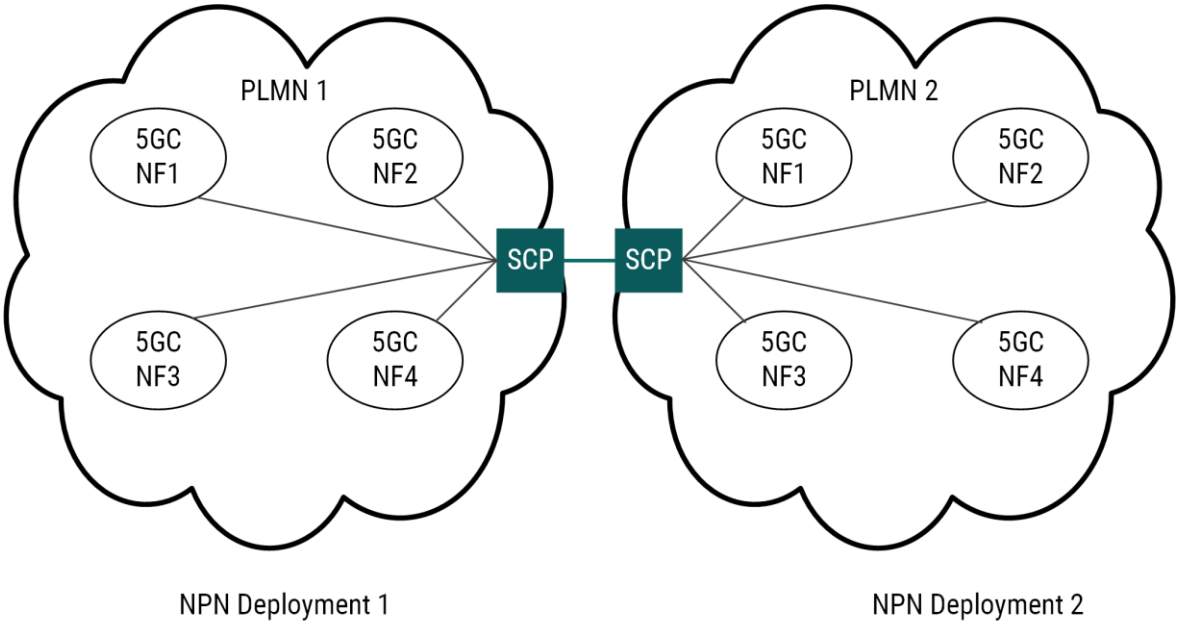

Figure 3.5: SCP-based inter domain routing

In case of Inter PLMN communications, the delegate discovery mechanism is used within the SBC to forward the message to the target NF. The NF consumer identifies the NF producer is not in the same PLMN and forwards the message to the local SCP with the discovery parameters such as- target NF type, target PLMN, target S-NSSAI etc. The local SCP (PLMN1) finds out the details of the SCP in the target domain (PLMN2) from its local cache and forwards the request to the remote SCP in the target domain (PLMN2). The SCP in the target domain (PLMN2) then queries the NRF (PLMN2) using the discovery parameters it received from the local SCP (PLMN1) to fetch the details of the target NF (PLMN2). Upon receiving the reply from the NRF, the remote SCP forwards the message to the target NF in that domain (PLMN2). The target NF (PLMN2) sends the response back to the NF consumer (PLMN1) through the remote SCP (PLMN2).

3.2.2 Service Routing on the User Plane

In today's 3GPP systems, including the latest releases that introduced 5G technology, there is a great deal of customization and pre-configuration. This pre-configuration and customization are intended to increase system efficiency, to e.g., execute node level O&M procedures to define virtual links rather than allowing the system to discover system components as needed, and route messages using cloud-based principles. Thus, one can find examples where AMFs and NG-RAN nodes are setup through a configuration procedure, e.g., linking a specific gNB with a specific AMF entity. The provisioning of 5G Core (5GC) network functions (NFs), in particular the SMF and UPF, are manually conducted. Unfortunately, this is done at the expense of flexibility and extensibility.

Furthermore, the 3GPP relies on the establishment of “data pipes” anchored to specific network entities, such as the User Plane Function (UPF), which once set up, must be used to transport data from the UE to the edge of the network or from UE to UE as is the case in 5GLAN type systems. These “data pipes” are referred to as PDU Sessions, are controlled by the 5GC which is composed of a set of dedicated NFs, responsible for establishing, maintaining, tearing down PDU Sessions, according to operator polices.

The integration of a service routing approach, such as Name-based Routing, to the user plane requires a carefully designed approach that a) allows the integration in the first instance and b) respects as many “Release 16 PDU session and node procedures” as possible to pave the way for a successful standardisation.

3.2.2.1 Base Assumptions

The solution presented here comes with a set of distinct assumptions, i.e.:

- The PDU session establishment, modification and release procedures on all nodes but the SMF remain untouched.
- Related to the first bullet point, no changes to NAS procedures by the UE and gNB are permitted to ease the standardisation and integration potential.
- Any additional NbR-related control plane interactions between UEs and 5GC to perform service routing for is kept in-band over already established PDU sessions.
- UPF-SMF’s N4 interface is upgraded to service-based semantics, i.e., HTTP/2 as the application layer protocol and the possibility to utilise the SCP to communicate with other NFs.

3.2.2.2 Infrastructure Mode

The proposed infrastructure mode allows UPFs to communicate via NbR procedures among each other, i.e. N9, while preserving the, in 3GPP, standardised system and procedures for N2, N3 and N19.

3.2.2.2.1 System Architecture

In order to achieve the integration of NbR into the 3GPP user plane, the two NbR management components PCE and SPM are exposed to the 5GC, as illustrated in Figure 3.6. For the PCE, which is responsible for rendezvous and FID calculation functionality, it is proposed to include this as part of the SMF via a dedicated Service-based Interface (SBI), Nsmf_NbR. This allows any consumer to utilise the functionality provided by the PCE, including other consumers that realise SMF functionality not covered by the PCE. Also, the architecture figure illustrates the N4 interface to become service based, Nupf, allowing any consumer to communicate with the UPF. For the integration of NbR the interface Nupf_NbR is proposed with dedicated Nupf primitives enabling the integration of name-based routing.

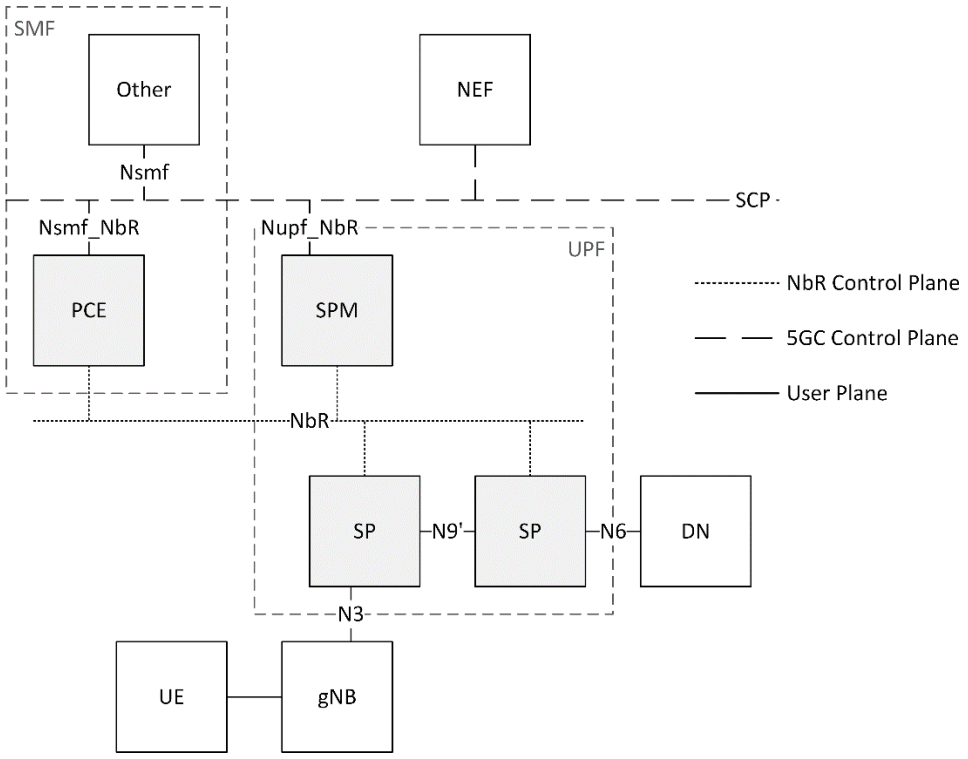

Figure 3.6: 3GPP Architecture for Name-based Routing on the user plane, infrastructure mode

The reason for the transition from N4 to Nupf is twofold: With 5GCs being orchestrated (automated deployment) in a cloud native fashion, any post-orchestration configuration of point-to-point endpoints (i.e. SMF and UPF on N4) becomes obsolete and the provisioning and communication towards the UPF follows the same cloud native principles as the other 5GC network functions (NFs). Secondly, Nupf allows more than one NF to act as a consumer and communicate with the UPF, finally removing any hard binding between two NFs from the 3GPP 5GC architecture.

Inter-UPF communication over N9 is defined as an IP-based communication in Release 16, but with the introduction of NbR for this interface, the architecture figure annotates this interface as N9', as NbR operates on top of 802.3.

For the NbR infrastructure mode, Figure 3.7 illustrates the protocol stack with NbR depicted as a layer on top of 802.3 for the N9' interface. The payload originating from the UE or DN can be of type IP or 802.3 (5GLAN) and are handled according to the NbR specifications for HTTP [TRO16] and non-HTTP IP-based [TRO15] communication.

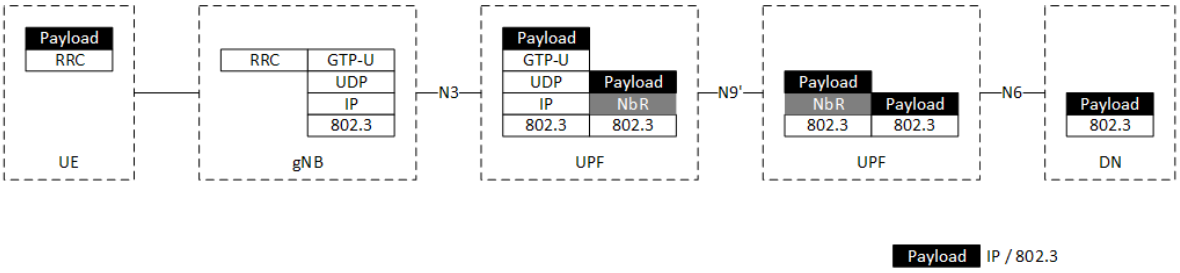

Figure 3.7: NbR user plane protocol stack for infrastructure mode for both PDU session types IP and 802.3

3.2.2.2.2 SMF and UPF Provisioning

The provisioning of SMF and UPF is foreseen as part of the automated deployment of a 5GC (aka orchestration) through an external technology such as OpenShift or Kubernetes integrated into ETSI MANO frameworks such as OpenSource Mano (OSM) or Open Network Automation Platform (ONAP).

Current procedures to provision SMF and UPFs are of rather static nature even when involving the NRF for these purposes. With the introduction of SBA and the advances of cloud native software design and engineering, such static provisioning must be avoided. As the targeted distributed UPF scenario requires topology management procedures (i.e. bootstrapping, link discovery and link failure detection), SDN (e.g. OpenFlow) is leveraged for this purpose. Figure 3.8 illustrates the procedures for UPF topology management with the integrated NbR system architecture provided in Figure 3.6.

1) The steps in here allow all NbR components to join an SDN switching fabric and to form a topology.
    a. All components that are attached to the NbR signalling plane (see Figure 3.6) leverage SDN for their communication among each other and connect to the SDN controller in this step.
    b. As part of standard SDN practices, the SDN controller requests all connected switches to discover their link-local neighbours via the Link Local Discovery Protocol (LLDP).
    c. The PCE component periodically queries the SDN controller via its northbound API.
    d. The SDN controller responds with the known topology which includes switch identifiers and its neighbours among other information. As northbound APIs for SDN controllers are not standardised the information provided for each switch can vary but always includes switch identifier and its neighbours.
2) The steps described in here demonstrate the message exchange for 5GC consumers with the PCE-enabled SMF and the newly proposed interface Nsmf_NbR. The steps hereafter describe a standard request response communication between a consumer and the SMF as the producer.

a. A consumer, e.g. another decomposed SMF service, requests an updated topology of UPFs using the `Nsmf_NbRTopologyRequest()` primitive.
b. The SMF producer serving the Nsmf_NbR primitives responds with an `Nsmf_NbRTopologyResponse()` which has the full topology. Note, only UPFs that implement the SP functionality and serve either N3, N6 or N9 interfaces will be returned in this primitive, as the NbR components PCE and SPM do not process user plane packets and therefore require no PDRs.
c. The consumer then requests the properties of a particular switch using the newly defined primitive `Nsmf_NbRUpfPropertiesRequest()`.
d. The producer responds to the request described in 2.c) with an `Nsmf_NbRUpfPropertiesResponse()` primitive which providers various properties for a particular switch ranging from port configurations. Most importantly though, the UPF properties provides the consumer with the information which routing technology a particular UPF implements.

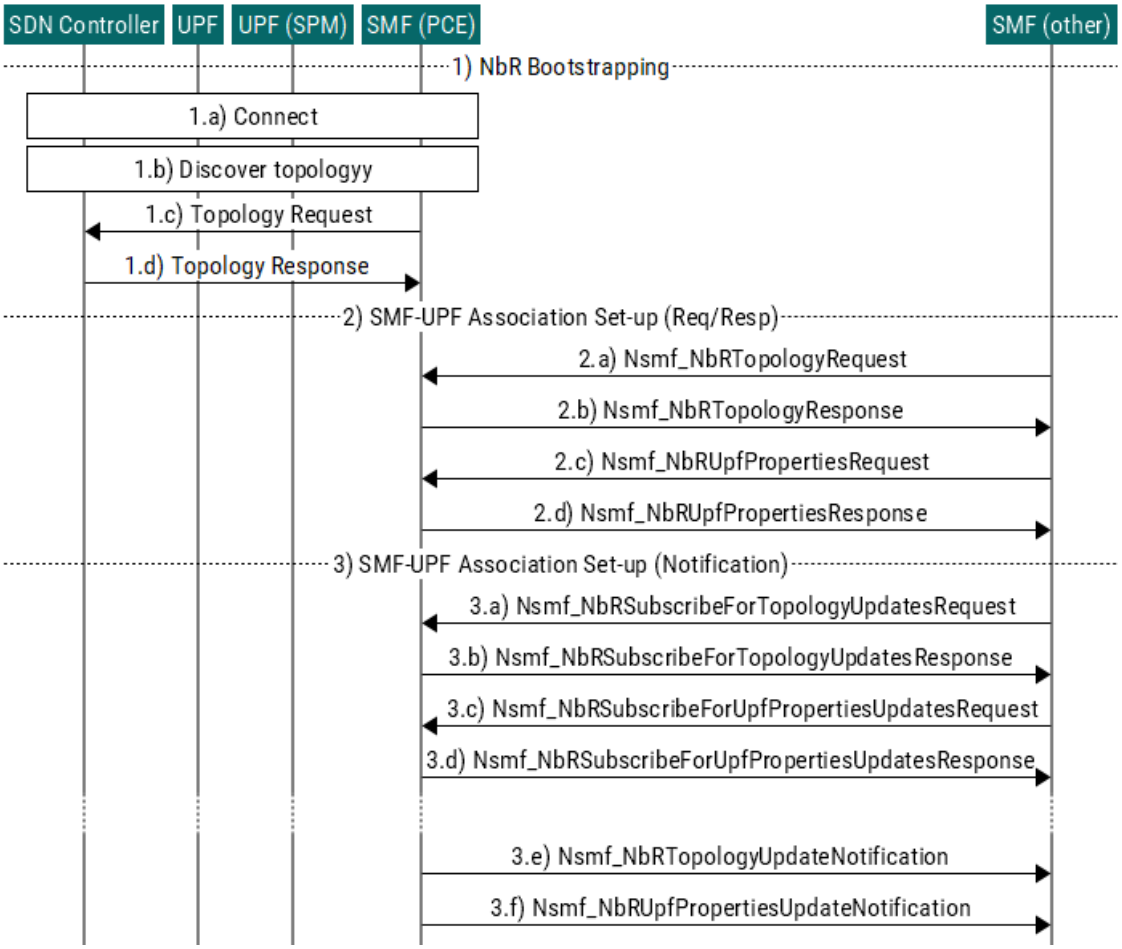

Figure 3.8: SMF and UPF provisioning

3) The steps described herein offer an alternative information exchange in comparison to the steps described in 2). For this purpose, a new set of primitives are being used.

a. A consumer, SMF, issues an `Nsmf_NbRSubscribeForTopologyUpdatesRequest()` to the SMF producer indicating how to communicate updates.
b. The SMF responds with an `Nsmf_NbRSubscribeForTopologyUpdatesResponse()` confirming the status of the notification registration.
c. A consumer, e.g. SMF, issues an `Nsmf_NbRSubscribeForUpfPropertiesUpdatesRequest()` to the SFM producer requesting to subscribe to future updates on any property changes for a particular UPF (SDN switch) from the set of UPFs (SDN switches) communicated in the topology.
d. The SMF responds with an `Nsmf_NbRSubscribeForUpfPropertiesUpdatesResponse()` confirming the status of the notification registration.
e. Once there is a topology update, the SMF goes through the list of subscribers to topology information and issues an `Nsmf_NbRTopologyUpdateNotification()` notification.
f. Once there is an update to the properties of a UPF, the SMF goes through the list of subscribers to UPF property updates and issues an `Nsmf_NbRUpfPropertiesUpdateNotification()` notification.

Upon completing the steps outlined above, the SMF has all information required to determine PDRs for future session requests by UEs.

3.2.2.2.3 Vertical Application Registration

NbR offers transparent service routing capabilities for stateless protocols, e.g. HTTP, allowing the switch of service endpoints (server) without affecting the requesting endpoint (client). In order to achieve that, the NbR layer executing the routing decisions as an UPF requires the information where service endpoints are located. Figure 3.9 presents the message sequence chart of how a vertical application located in an DN is registered against NbR-based UPFs.

1) The NEF receives the information about the existence of a new vertical application in a particular DN following standard Release 16 procedures.
2) The NEF now uses the newly designed `Nupf` interface to communicate the existence of the new vertical application to the UPF using the `Nupf_NbRFQDNRegistrationRequest()` primitive.
3) The UPF_{SPM} distributes the information across all SPs that implement the user plane packet routing functionality. This is exchanged over the NbR internal pub/sub-based control plane.
4) Upon distributing the registration information, the UPF_{SPM} confirms the status of the registration to the consumer (NEF) using the

`Nupf_NbRFQDNRegistrationResponse()` primitive. This primitive follows the methods and procedures disclosed in [ROB171].

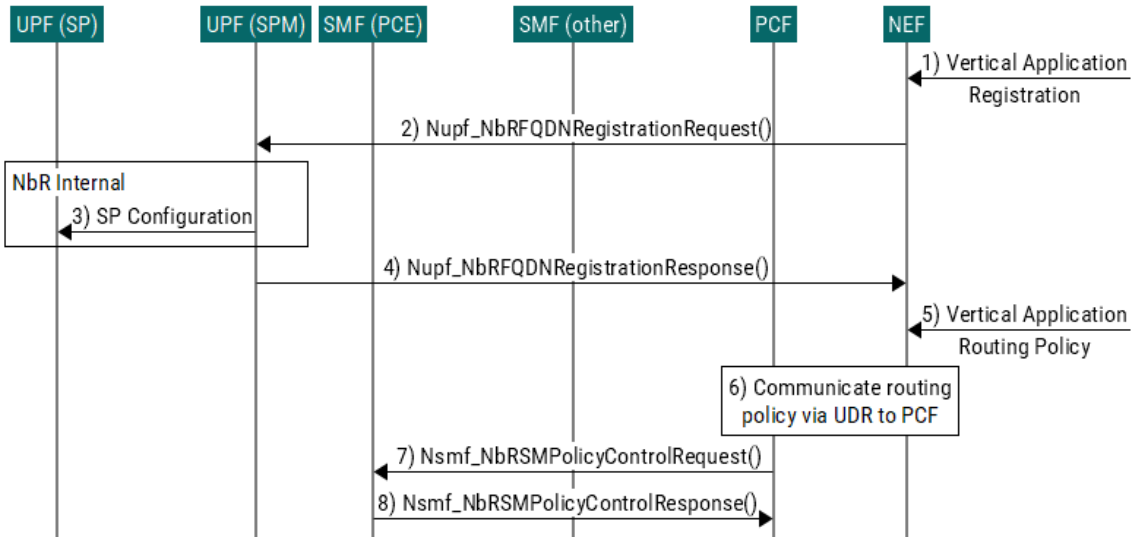

Figure 3.9: Registration of vertical application against NbR UPF

5) Following Release 16 specification, the NEF receives the request for specific routing policy instructions (e.g. from an AF).
6) Together with the UDR and UDF, the PCF establishes the appropriate routing policy.
7) The outcome of the desired routing policy is communicated to the SMF_{PCE} via the primitive `Nsmf_NbRSMPolicyControlRequest()`.
8) The SMFPCE consumer responds with a Nsmf_NbRSMPolicyControlResponse() indicating the processing state of the policy request.

3.2.2.2.4 Session Establishment

This section describes the procedures for the establishment a PDU session over an NbR-based user plane in infrastructure mode as well as the actual uplink and downlink data exchange. Figure 3.10 provides the message sequence chart for the procedures. Note, the steps in this section cover both IP-based and 802.3-based PDU session types and starts with the communication of a session establishment request by an SMF either as part of a proactive UPF configuration or as part of a PDU session establishment procedure.

1) The SMF communicates the Packet Detection Rules (PDRs) for the UPFs involved using the `Nupf_NbRSessionEstablishmentRequest()` primitive and covers the information provided by the N4_SessionEstablishmentRequest primitive, as described in Release 16. The difference is that the consumer (SMF) must specify which distributed UPF is receiving which set of PDRs utilising the information obtained during the provisioning procedures. This information is known to the SMF from the provisioning procedures described in Section 3.2.2.2.2 where the UPF properties define the NbR mode an UPF implements (infrastructure or UE).

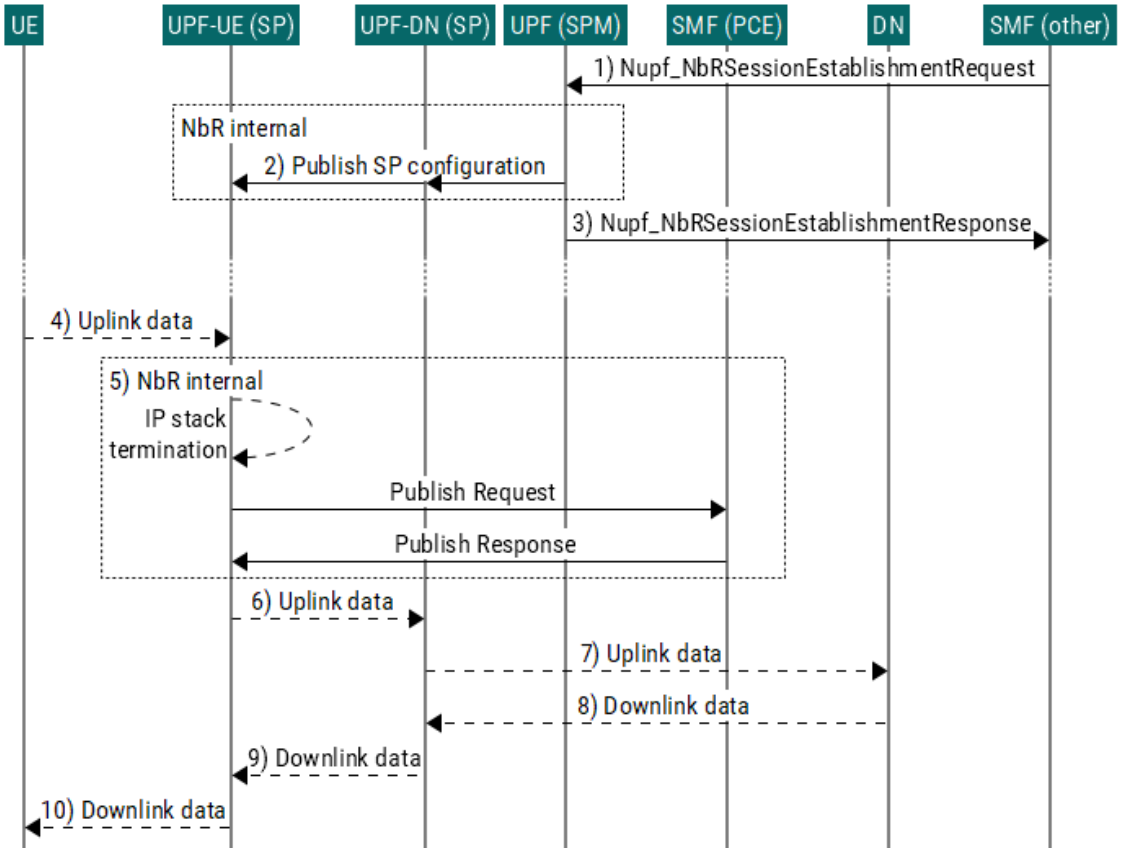

Figure 3.10: Session establishment for NbR UPFs operating in infrastructure mode

2) Using the NbR-internal signalling pub/sub system, the Nupf information for each UPF is communicated to all SPs.
3) Using the `Nupf_NbRSessionEstablishmentResponse()` primitive, the UPF_{SPM} informs the consumer (SMF in the chart illustrated in Figure 3.10) about the status of the session establishment request from Step 1).
4) Following Rel.16 procedures, the UE eventually receives the conformation that the PDU session has been stablished and is being able to send data over the user plane which arrives at the UPF-UE component.
5) The UPF-UE then applies the Name-based Routing methods and procedures to translate IP into ICN, as disclosed in [TRO15], [ROB172], [HER19], and/or [TRO181] depending on the traffic that arrives, i.e. TLS, HTTP or any other IP-based communication. As part of these procedures, the UPF-UE communicates over NbR with the SMF_{PCE} in order to perform the ICN semantics, i.e. rendezvous, and path calculation informing the UPF-UE about the UPF-DN which will be able to serve the request received by the UE.
6) The UPF-UE sends the uplink data to the UPF-DN where it is translated, transparently for both IP endpoints, into a standard IP-based communication.
7) The uplink data is being sent to the DN where a vertical application will process the request.

8) Upon the generation of the response to the received request by the UE, the vertical application in the DN issues the response using a standard IP-based communication stack.
9) The response (aka downlink data) is being translated into ICN by the UPN-DN and being sent over an L2 switching fabric to the UPF-UE.
10) The UPF-UE then translates the downlink data back into a standard IP-based communication and sends it to the UE via a standard N3 realisation.

3.2.2.3 UE Mode

For scenarios where the PDU session type is Ethernet and the UE sends 802.3 packet headers with payload on the user plane towards the UPF, this section describes the integration of NbR and its support for 5GLAN into a 5GC.

3.2.2.3.1 System Architecture

Figure 3.11 illustrates the system architecture for an 5G core integration based on the infrastructure mode with only two changes:

- The procedures for NbR over 5GLAN were previously disclosed in [TRO19] and describes the SP residing on the UE where IP traffic is being translated into ICN and vice versa. The SP then communicates over a standard 802.3 frame header with the UPF. As part of the NbR header in the payload of the ethernet header, [TRO19] describes the additional fields it would add when another UE is the destination.
- While moving the SP functionality into the UE, the path-based forwarding approach NbR is based on where the PCE calculates the path through the network based on the pub/sub decision of the PCE which SP is supposed to be addressed. While the procedures for requesting a FID reside within the SP, the communication between UEs and UPFs are not part of this FID. Therefore, a Service Proxy Forwarder (SPF) is being illustrated in Figure 3.11 as part of the UPF and performs UPF functionalities according to the N3 interface specification. The SPF component implements the counterpart of the UE procedures disclosed in [TRO19].

In summary, the architectural changes only affect the UE and UPF communicating to a gNB/UE. All other components and interfaces remain identical to the infrastructure mode described above.

However, while [TRO19] describes the signalling for NbR internal pub/sub semantics as an extension to the PDU session establishment via NAS, the procedures and methods proposed in this work follow an “in-band” NbR control plane communication over an established PDU session (user plane) in order to enable the support for Release 16 compliant UEs and gNBs. This will be described in further detail in Section 3.2.2.2.2.

Similar to the infrastructure mode, UPFs that implement either the SP or SPF communicate with each other over 802.3 and the interface is therefore annotated as N9’.

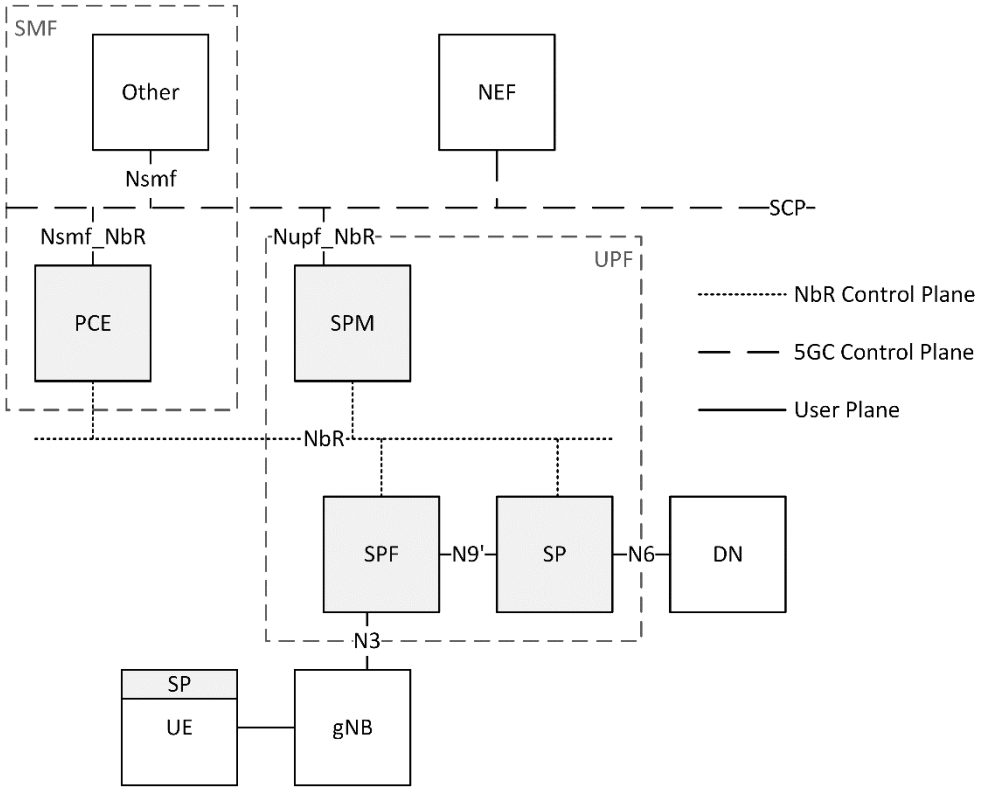

Figure 3.11: 3GPP Architecture for Name-based Routing on the user plane, UE mode

Figure 3.12 illustrates the user plane protocol stack for the UE mode. With the NbR layer extended to the UE, the UE mode only supports the PDU session type Ethernet. The payload can be any IP-based protocol with NbR offering special service routing capabilities for HTTP (including TLS-based HTTP communication).

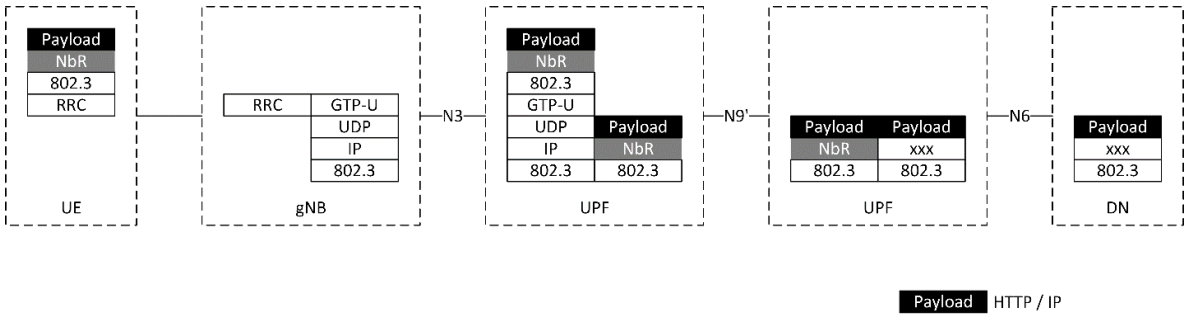

Figure 3.12: NbR user plane protocol stack for UE mode over 802.3

3.2.2.3.2 Session Establishment

In principle, the session establishment for the UE mode follows the same assumptions as in the infrastructure mode, i.e. UPF configurations for N3 are communicated via the Nupf interface by SMF while the required pub/sub communication for finding the most appropriate service endpoint that will receive the packet is communicated in an in-band fashion.

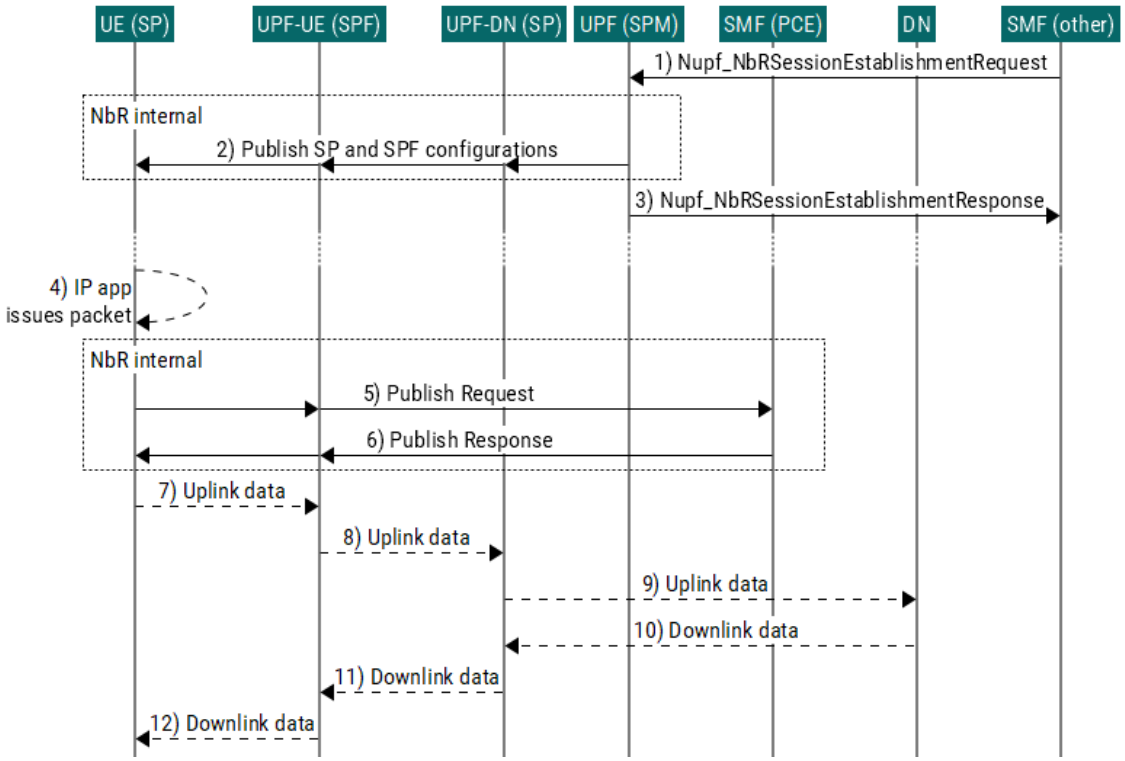

Figure 3.13: Session establishment for NbR UPFs operating in UE mode

1) The SMF communicates the PDRs for the UPFs involved using the `Nupf_NbRSessionEstablishmentRequest()` primitive and covers the information provided by the N4_SessionEstablishmentRequest primitive, as described in Release 16. The difference is that the consumer (SMF) must specify which distributed UPF is receiving which set of PDRs utilising the information obtained during the provisioning procedures. In the UE mode, the UPF facing the gNB via N3 is the SPF which requires the MAC address of the UE for which a PDU session will be/is being established.
2) Using the NbR-internal pub/sub signalling system, the N4 information for each SP and SPF is communicated by the SPM. For SPs located on UEs, the SPF acts as the forwarder for information related to NbR control plane procedures, e.g. where to find the PCE (in terms of the FID) for future publish requests. As the PDU session is has not been configured in the gNB or UE at this moment of time, the SPF will not be able to reach the SP on the UE yet. It is foreseen that the SPF either periodically communicates the SPM information to towards the UE until it explicitly acknowledges it. In another embodiment, the SP on the UE awaits the PDU session to be established and only then communicates with the SPF via a broadcast 802.3 frame that it requires any SPM information which the SPF holds.
3) The UPF_{SPM} responds top the consumer that requested the session establishment in 1) with an `Nupf_NbRSessionEstablishmentResponse()` primitive.

4) Once the application on the UE issues an IP-based communication the SP transparently intercepts any transaction according to the steps described in [HER19], [ROB172], [TRO15] and [TRO181].
5) The SP issues a publish request towards the PCE using the Ethernet PDU session towards the UPF_{SPF} with the FID provided by the SPM in Step 2). The UPF_{SPF} then forwards the request to the SMF_{PCE}.
6) The SMF_{PCE} response with the decision about the request. If a subscriber exists for the IP address or FQDN, the response comprises the information the UE_{SP} requires to reach the destination (inventions [HER19], [TRO15] and [TRO181] apply).
7) The UE_{SP} now sends the uplink data via the Ethernet PDU session towards the UPF following the protocol stack illustrated in Figure 3.12.
8) The UPF serving the UE (UPF_{SPF}) forwards the packet to the UPF that serves the DN where the IP service endpoint resides that can handle the transaction by the UE.
9) The SP in the UPF-DN translates the payload received by the UE back to an IP-based communication and sends off the transaction to the DN.
10) Upon processing the request, the IP-based application inside the DN responds with an IP-based transaction which is sent to the UPF-DN.
11) The UPF-DN then uses the methods and procedures in [HER19], [ROB172], [TRO15], [TRO181] and/or [TRO182] to transparently translate the IP-based traffic into an ICN communication. Following the protocol stack illustrated in Figure 3.12, the downlink data is being sent to the UPF-UE.
12) The SPF in the UPF-UE now applies the methods and procedures disclosed in [TRO19] to reach the UE over a 5GLAN PDU session.
13) Upon reception of an NbR communication, the SP inside the UE translates the ICN communication transparently back into an IP-based.

3.2.3 Resource Scheduling

The task of the resource scheduling is to determine the most suitable CNF instance for an incoming request at run-time. Herein, the objective could be, e.g. a simple load distribution across several CNF instances, but could also be much more complex, e.g., reliability increase in some failure model, shortest overall service execution delay or least overall energy consumption. Accordingly, the actual selection of the CNF instance could be as simple as random (randomly select an instance from a known instance list), greedy (e.g., topologically closest instance to the ingress node) or driven by service- and user-specific metrics, such as service quality budgets or directing a specific user to a specific service instance based on user information such as the SUPI, previous sessions and similar.

At the first glance, one could think of solving the resource scheduling problem above through a pre-provisioning approach, i.e., with suitable resource provisioning and appropriate configurations at the deployment time. Indeed, presuming *a priori* knowledge of a) the types of traffic, b) associated loads and c) traffic distribution (geographically and

in time), one should be able, in principle, to conceive and deploy a system optimally organised to serve that particular traffic mix. Alas, the reality of the modern system deployments is rather radically different from this "ideal pre-provisioning" idea. As illustrated by the current trends to softwarised and flexible systems and networks, an optimal fixed-shape system idea has been mostly abandoned; it has become an almost utopic idea. The reason for that is the huge variety of applications and of the corresponding requirements of different stakeholders, mobility of both users and applications and the strongly and unpredictably fluctuating service demands and traffic distributions, which together make it almost impossible in practice to get the required *a priori* information. Besides, facing tight guarantees, fixed pre-provisioning approaches are known to be resource-inefficient, as they tend to strong over-provisioning (to be able to serve the load spikes correctly). This therefore raises additional questions with regard to the economic and ecological sustainability of the overall approach.

For this reason, the essence of the resource scheduling component treated here is the runtime approach, as first introduced in [BLO20] and not a deployment time / dimensioning approach like, e.g., VNE and the like. We assume that some dimensioning / pre-provisioning can be done depending on the particular stakeholder and deployment context and work within the resulting system. Existing access and admission and control policies, albeit relevant to the question of the overall system load, are understood as orthogonal to the question discussed in this section: we presume that access control filters out all unauthorized traffic, and, very similarly, we presume that any admission control decisions are done (by state of the art means) prior to the activation of the runtime resource scheduling component. In other words, all traffic treated by the resource scheduling component is legit and needs to be served correctly. However, no technical component can avoid all overloads or failures. In summary, this means that, ultimately, **the task of the resource scheduling component is to maximize the probability that each incoming request is served within the associated service policy** (e.g. delay or power budgets, quotas, max. loads, etc.), including the resource consumption of the resource scheduling component as such.

As a runtime approach, the resource scheduling component requires interfaces to other system components, notably to get the required incoming traffic constraints, a list of the suitable CNF instances in the system and the state information (availability, load, remaining resources, etc.) of the latter instances.

Ideally, all such interfaces would deliver instantaneous, fresh data. Also ideally, and in a small setup or as a prototype, the resource scheduling component could be conceived of as a single central element getting all incoming traffic requests with constraints, gathering all semantically suitable and available CNF instance information (in particular, availability and load information) over those interfaces for provisioning the necessary input into the

decision-making and, finally, selecting exactly one treating instance as per optimization target.

Realistically however, given its runtime aspect and the presumed system size, we should conceive of resource scheduling as of a distributed mechanism with a distributed protocol running within the infrastructure. To account for well-known distributed system artefacts, it should be prepared to work with only incomplete and possibly outdated state data. Hence, in terms of suitable design space, solutions similar to routing protocols may be used for conveying the required CNF instance states for the decision making, similar to solutions in [SAV16]. Due to the nature of this specific input gathering, we leave the precise realization of most interfaces and corresponding protocols to a latter phase (notably to the realization phase of this component), while noting the necessity for such interfaces at an abstract level.

For the high-level architecture in Figure 3.1, the most important interface for the Resource Scheduling is the one to the Service Routing component, i.e. Sx (fully specified in the final architecture deliverable), realising the Item 4 in the introductory list of capabilities being realised by the overall Routing component. The requirements for this interface are as follows:

- This interface MUST initiate the selection process.
    - The request MAY include a set of suitable CNF instance identifiers for said selection.
- As the response, the resource scheduling MUST provide a single CNF instance identifier over the same interface.
    - Said identifier MUST fall within the provided set of suitable CNF instance identifiers that MAY have been given as part of the request.

In other words, even though the objectives of Resource Scheduling could be expressed in very general terms and try to reach different targets, in FUDGE-5G, the main objective of resource scheduling component is linked to the Routing component of the high-level architecture. Accordingly, in FUDGE-5G, the chosen implementation is linked to the Service Routing and is positioned at the network level.

The proposed Compute-Aware Distributed Scheduling (CADS) is realized as a distributed system of schedulers, which perform service-specific scheduling at runtime at the routing level. In order to support better integration with the service routing component, the scheduling decisions are performed independently at each individual semantic router positioned at the network ingress, which is the SCP in SBA, and the service request forwarding can be executed for ingress-egress architecture as required by SBA. The semantic routers can forward requests to the egress point based on semantic identifiers and the available path information at the SCP. CADS allows for taking into consideration service-specific constraints for scheduling decisions, not limited to network and service

constraints, and supports longer-lived transactions (flow affinity), i.e. ensuring that subsequent requests of the same transaction are routed to the same CNF instance. The initial implementation of the compute-aware scheduling approach allows for more efficient use of compute resources, which supports in the reduction in the service completion delay, while the system model allows for considering other constraints based on the aforementioned resource scheduling objective.

3.2.4 Control vs User Plane Considerations

In both the resource scheduler concept and its current implementation (work in progress), there is nothing that is exclusively control plane (CP) or user-plane (UP) specific. Instead, we apply the Service Function chaining logic by assuming that each request needs to traverse (or be "routed" through) a number of semantically different processing stages. The overall desired effect is assumed to be achieved, when all required stages have been successfully (i.e. within the quality constraints, e.g. delay budgets, etc.) traversed. Both 5GC CP and UP can be modelled this way, and both CP and UP processing chains can be reasonably well associated with required performance parameters, e.g. maximum overall delay of execution (e.g. until request expiration or until QoS underachievement, etc), justifying the need for a potentially more intelligent choice of processing locations (NF instances) for each of the stages. The main changes between the CP and UP would be in the question of the originator and the ingress point respectively (e.g. a gNB or an AMF instance for the CP, and a user and UPF for the UP).

Precisely, the general scheduler logic per se does not really care, what runs in the NF instances (e.g. if it is an NWDAF instance or a video encoder/decoder, or anything similar, on the UP). The current implementation does look more at the CP issues, as the scheduler instances would be deployed on the SCP instances, which is rather CP typical at the moment. However, scheduler instances could be deployed within the UPF controller or even within the SMF, to choose between equivalent UPF pipelines.

3.3 Service Orchestration and Telemetry

FUDGE-5G aggressively advocates for the transitioning from VNFs to CNFs for 5GC NFs and vertical applications. Such a paradigm shift does not only allow the adoption of well-proven web technologies for the realisation of an application (aka microservice software architecture), it also enables a unified cloud native orchestration of a service. As illustrated in Section 3.1, the FUDGE-5G system architecture comprises an orchestration layer focusing on the orchestration and lifecycle management of 5GC NFs and vertical applications.

3.3.1 Service Function Virtualisation Orchestrator

Figure 3.14 zooms into the orchestration layer and illustrates the individual components and their interfaces using UML syntax highlighting endpoints (client) and service endpoints (server). As can be seen, the Vertical Application Orchestrator (VAO) is logically located above Service Function Virtualisation (SFV). This is mainly due to the objective to unify advances of the FUDGE-5G system where the orchestration layer itself is agnostic to the microservice that is being orchestrated and lifecycle managed, i.e. 5GC vs vertical application. Both services are a composition of service functions that form a service chain following advances of Service Function Chaining (SFC). As SFC is concerned about the routing and its configuration among service functions of a service chain inside the routing fabric, FUDGE-5G describes the required evolution of NFV for microservices as Service Function Virtualisation (SFV). It is worth noting that SFV follows the ETSI MANO reference model and can be seen as a counterpart to the recently published ETSI IFA 040 specification [ETS20] aiming for a reference model for CNFs.

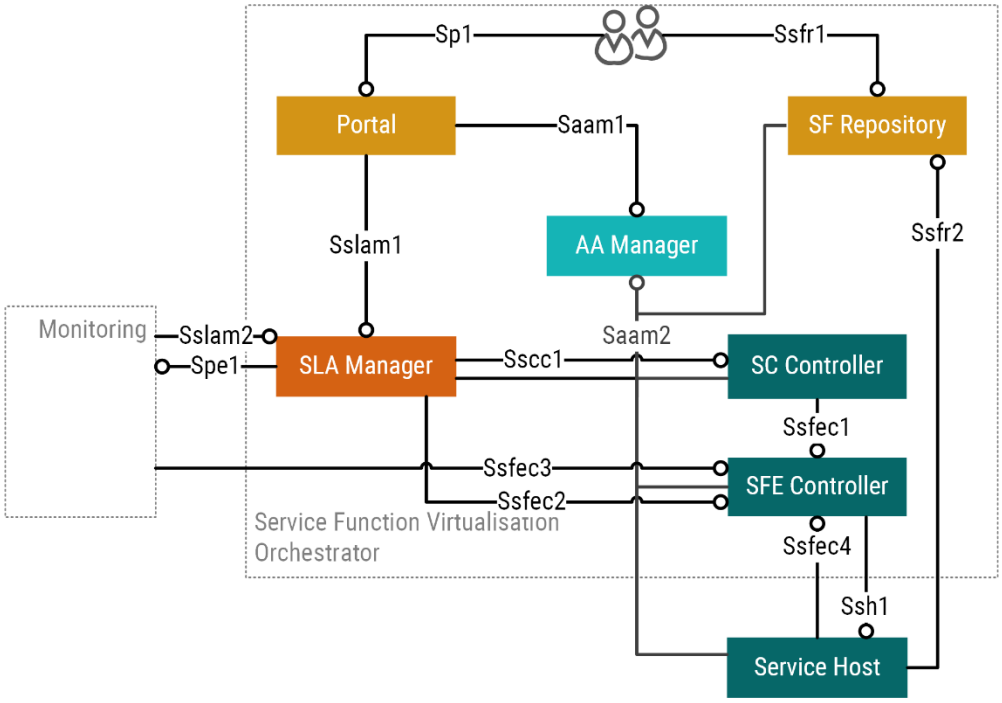

Figure 3.14: Component architecture of the FUDGE-5G Service Function Orchestration and Monitoring layer

Figure 3.14 illustrates the component and interface architecture of the cloud native orchestration layer inside the FUDGE-5G platform (see Figure 3.1). This layer is composed of the Vertical Application Orchestrator (VAO) and SFV Orchestrator (SFVO). While the concepts around SFV are being used for both the orchestration of a 5GC as well as the orchestration of a vertical application, the VAO – as the name implies – only provides its functionality for verticals and can be seen as additional functionality provided by the FUDGE-5G platform to unify certain aspects of the workflow. The figure above provides the names of all service-based interfaces a component offers following the same naming

convention, i.e. *S* for Service followed by a *lower-case letter* identifying the component that provides the interface and an *integer number* unique to the component.

The SFV Orchestrator (SFVO), composed of the components drawn within the SFV layer in Figure 3.14, follows an information model which is derived from Service Function Chaining (SFC) in its terminology. This model, illustrated in Figure 3.15, is categorised into orchestration, lifecycle management, routing and packaging.

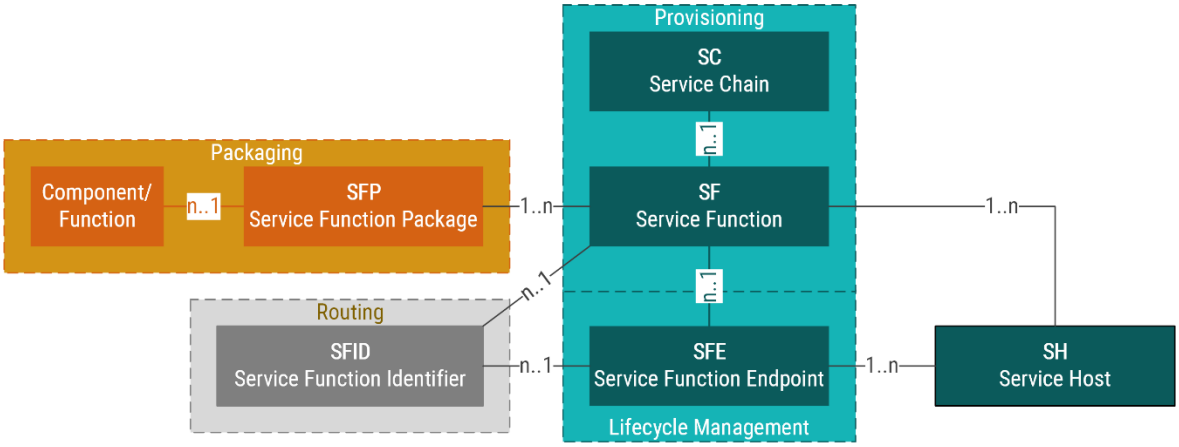

Figure 3.15: SFV information model

Provisioning and Lifecycle Management

The service that is orchestrated is declared as a **Service Chain** (SC) which is essentially an arbitrary but – within the tenant's orchestration slice – unique name, allowing for its identification. Each service chain then has one or more **Service Functions** (SFs) that represent the actual decomposed application the service chain embodies. Each SF is then represented by a unique **Service Function Identifier** (SFID) (e.g. fully qualified domain name (FQDN)) and linked against the routing layer for registration of the identifier (more information in the paragraph below about routing). SFs are then orchestrated as instances of SFs, called **Service Function Endpoints** (SFEs). Note, the SFV information model also allow the assignment of an SFID against a subset of SFEs representing the same SF with the subset in the range of *{1..n-1}* with *n* being all SFEs of the same SF; SFE.

SFEs are orchestrated into a specific state across **Service Host** (SH) which represent compute devices capable of hosting the SFE. For instance, an SH can be a larger VNF that maxes out the compute, networking and storage properties provided by the infrastructure provider on a compute node or any other host such as UEs. The lifecycle states offered by SFV are:

- NON_PLACED: The packaged service function is logically accounted on the cluster but does not consume any physical computing resources (vCPU, memory, storage).

- PLACED: The packaged service function is placed on the cluster and is logically accounted against the available resources. Thus, physically it only consumes storage but no vCPUs or memory.
- BOOTED: The service function is placed and started on the cluster but has not been registered against the platform and therefore is not reachable by any service. However, it allows the bootstrapping of all SF internal components.
- CONNECTED: This state registers the service function identifier (FQDN) against the platform and is reachable by any service under the given identifier.

With the information model presented above, an extended set of SF scaling scenarios are enabled through the ability of location-aware orchestration with the inclusion of the Service Host into the information model. SFV supports typical vertical (change properties such as number of vCPUs), horizontal (increase number of instances) and global scaling scenarios (increase number of locations where service is offered).

Routing

The routing of packets among SFEs is decoupled from the orchestration layer and is independent from which routing technology is being used (IP, NbR, etc). However, it is expected that the routing layer offers the features described in Section 3.2.

At orchestration time, the SFIDs for each Service Function are communicated to the routing layer through a registration interface. The routing layer then determines at run time which SFE to choose for any occurring request at an ingress point to the data plane.

Packaging

Packaging up an SF into a Service Function Package (SFP) essentially results in an image for a particular hypervisor (e.g. KVM or Xen), virtualisation technologies (e.g., LXC, Docker, rkt, Kata or UniKernel) or native system (APK for Android, IPA for Apple, EXE for Windows or deb for Debian-based systems) which can be imported and spawn up. While an SFP can host more than one function/component, cloud-native deployments strongly demand the creation of microservices to allow an orchestration down to function-level if desired. If more than one function is packaged into an SFP the SFVO can only orchestrate SFs and lifecycle manage SFEs, but not the components inside an SFE.

In order to describe the service chain for the various components that form the ability of provisioning and LCM, Figure 3.16 illustrates the SFV descriptor model. The model is colour following the orchestration layer architecture provided in Figure 3.14.

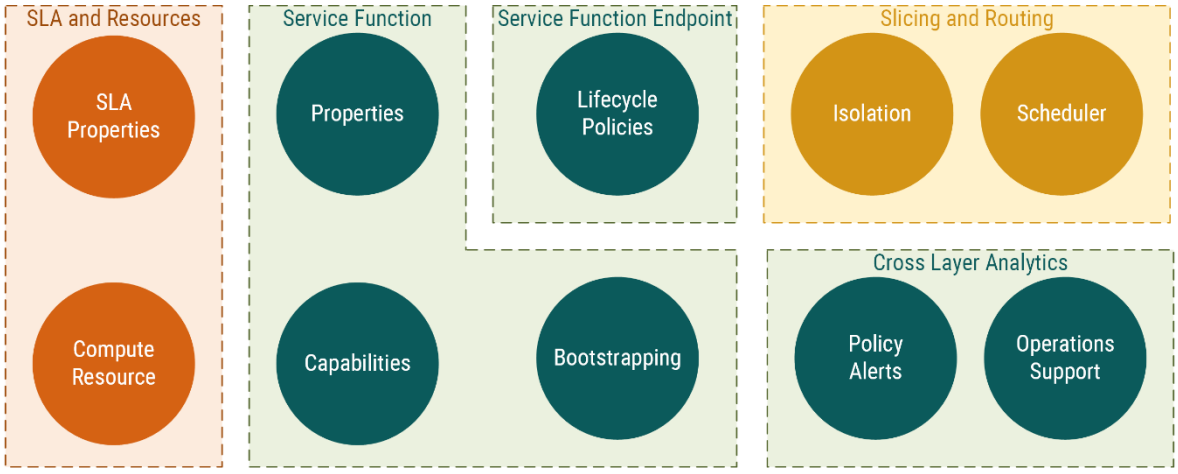

Figure 3.16: SFV descriptor model

SLA Properties and Compute Resource are descriptors defining the required compute resources for the service chain, broken down by SFs. This functionality resides within the VAO layer.

Service Function offers the ability to describe the properties of a service function and its capabilities as well as required instructions at bootstrapping time that are service specific.

Service Function Endpoint is a descriptor defining the policy triggers names and LCM state changes per service host.

Slicing and Routing allows the configuration of isolating a service chain from communication originating from any IP endpoint that is not part of the Service Chain. Additionally, the scheduling behaviour can be configured for SFEs served by the same egress point of the routing layer.

The **Cross Layer Analytics** set of descriptors allow the configuration of LCM triggers (policy alerts) based on continuous KPI monitoring and analytics as well as a wide range of operations support actions, e.g. security-related alerts about DoS attacks or potential malicious behaviour.

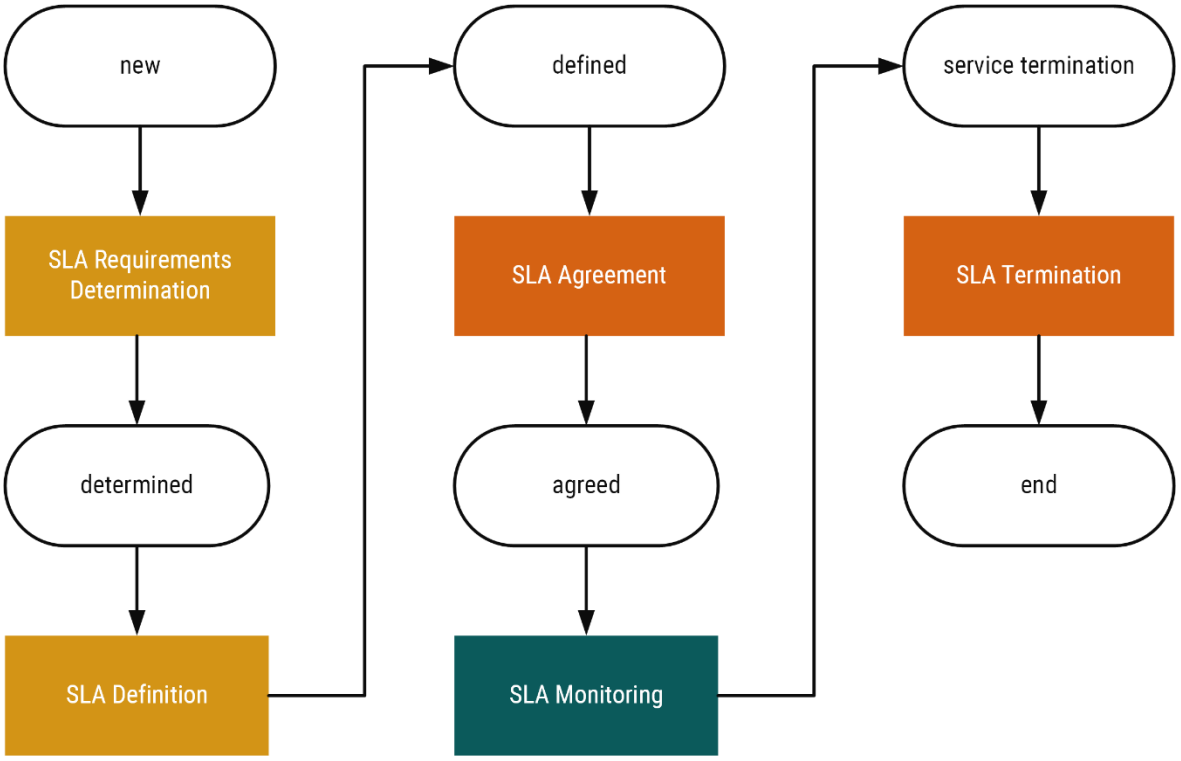

Figure 3.17: SLA business logic for orchestrating Service Chains.

The SLA negotiation procedures adopt the business logic outlined in ITIL4 [ITI19] which is illustrated in the FUDGE-5G context in Figure 3.17. Following the same colour schema, as in the orchestration architecture Figure 3.14, the SLA requirements determination and definition is being handled by an authoritative entity in a northbound fashion with the SLAM, i.e. the 5GC vendor or the VAO. Upon the definition of the SLAs, they are being communicated to the SLAM which in turn determines the most optimal placements (location) and density (number of SFEs per location) of SFs including monitoring policies and LCM triggers to cope with changing KPIs to meet the SLAs. Upon the explicit SLA agreement of the resource and policy descriptors, continuous SLA monitoring (conducted by CLA) and LCM (executed by the SFEC) ensure the validity of the agreed SLAs.

3.3.1.1 Authentication and Authorisation Manager

The Authentication and Authorisation Manager (AAM) providers a unified user management component as part of the SFV layer to SFV and VAO components. The need for such unified component can be explained by the necessity for each SFVO or VAO module to understand the relationship of users to resource requests. Not only does this allow to work on the same set of tenants and their credentials across the components, it feeds directly into the ability for an end-to-end (north-south and east-west) system slicing.

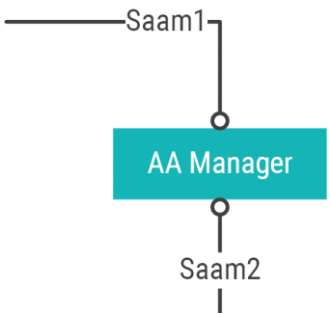

Figure 3.18: Interfaces of the Authentication and Authorisation Manager

The **Saam1** interface of the AAM allows an endpoint to request the creation of a temporary token based on user credentials and a common shared secret. This token is then used in each inter component transaction allowing service endpoints to verify that requests are indeed issued from genuine endpoints.

The **Saam2** interface adds the ability for any component to request the verification of the user credentials themselves allowing an additional layer of security for scenarios where the endpoint and service endpoint are not orchestrated as part of the same platform.

3.3.1.2 Service Level Agreement Manager

The Service Level Agreement Manger (SLAM) is concerned of choosing the most suitable Service Function Packages (SFPs) forming a Service Function (SF), while meeting Service Level Agreement (SLA) requirements of corresponding individual Service Functions (SF) and the overall Service Chain (SC). The SLAM utilises a machine learning-driven approach to determine the optimised placement of SFs across available Service Host (SHs) taking given SLAs into account. The output of such operation is a resource orchestration descriptor which is being communicated to the SCC for execution.

Runtime monitoring information of the environment are periodically gathered for observing the performance of the management decisions being made, and SF and connectivity selection procedures may be re-executed if any improvements to the current performance can be made, through the selection of new SFs and corresponding SFC configurations. Moreover, in case where a learning algorithm is used, such runtime monitoring information may be used for training the algorithm.

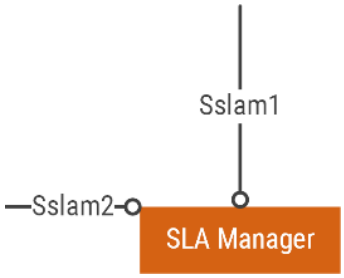

Figure 3.19: Interfaces provided by the SLAM component

The **Sslam1** interface allows the communication of the service chain, its service functions and their required SLAs. Following ISO 20000, the outcome of such request is presented back to the requesting entity for explicit agreement of the resource and policy descriptors. Upon the agreement, the SLAM receives the request to execute an orchestration and the instantiation of the required monitoring policies for lifecycle management purposes.

The **Sm2** interface allows the communication of an alert that all policies have been exhausted to meet the SLAs and no other LCM policy is available. As an outcome of this alert, the SLAM might re-orchestrate the service chain.

3.3.1.3 Service Chain Controller

The Service Chain Controller (SCC) manages and abstracts the overall deployment of virtual instances on the available service platform. Within the SCC a deployment specification template parser performs a validation at syntactical and semantical level against the Service Function Virtualisation Domain Specific Language specifications. The validator is the first unit which receives deployment requests from either the Portal or the SLA manager. Via the portal, the SCC enables the service and core network providers to push and modify the resource specification of the given use case, scenario or core network deployment. The given document from the portal or SLA manager describes the SFC with its SFs and its targeted lifecycle management.

Another sub-component of the SCC is the maintainer of the information model for the infrastructure resources. This information is used from the validator and parser to evaluate the resource requirements of the deployable SFs on each of the targeted Service Hosts of the platform.

Within the architecture itself, it is not foreseen to have a WebGUI for convenient upload or modification of the deployments. Hence, it is a machine-to-machine only interface. The interface itself is implemented in a HTTP-RESTful way, leveraging proper Request/Response status treatment and further detailed resource information.

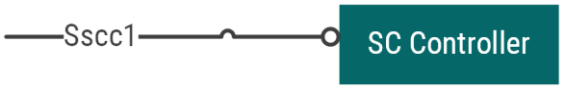

Figure 3.20: Interfaces provided by the SCC component

The SCC offers one interface. **Sscc1** allows to communicate the description of the service chain using a resource descriptor definition similar to TOSCA or Heat Orchestration Template (HOT) descriptors. More information on this descriptor can be found in FUDGE-5G's D2.1 [FUD21].

3.3.1.4 Service Function Endpoint Controller

The Service Function Endpoint Controller (SFEC) It consists of three elements: the Service Function Control, the Virtual Instance Manager and the Service Function Monitor. The

Controller itself is utilized to allocate resources of deployable Endpoints per host, based on the given resource specification, the required container images of an SF, as well as on the number of possible replications on the available hosts of the service platform. The Instance Manager maintains the lifecycle of each SF Endpoint. Based on given instructions within the deployment specification or from policies, the Instance Manager sets the SFEs into the targeted lifecycle state (either Non-Placed, Placed, Booted, or Connected). Each SFE passes certain states within this state machine. This is monitored by the Service Function Monitor which keeps track of the resource consumption within the service platform for each service host and offers this information as transaction state to the requesting unit.

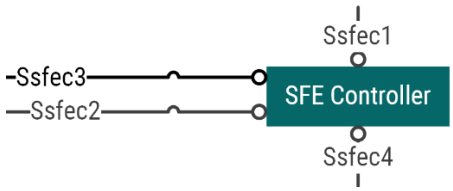

Figure 3.21: Interfaces provided by the SFEC component

The SFEC providers four interfaces. **Ssfec1** offers the ability to instantiate a new SFE into a given state. **Ssfec2** allows the retrieval of registered service hosts and their capabilities as well as the currently utilisation of said capabilities. **Ssfec3** offers the ability to trigger policies to change the state of existing SFEs. **Ssfec4** allows Service Hosts to register against the orchestration layer by communicating their capabilities such as supported virtualisation technologies, compute, networking and storage resources.

3.3.1.5 Service Host

The Service Host (SH) allows to run and maintain the containers locally deployed on the host's virtual infrastructure. The Host runs therefore a local SFEC Agent. This allows that the host's environment is part of the overall service platform and its available (local) resources are abstracted and represented within the platform-wide resources and capabilities. Such Service Hosts are not limited to COTS hardware. Hence, within the project containers may also run on mobile operating systems (such as Android). This allows an expansion of a controllable service platform much closer to the customer.

Figure 3.22: Interfaces provided by the SH component

An SH provides one interfaces, **Ssh1**, which allows to submit instructions to create, manage and delete SFE on a service host.

3.3.2 Vertical Application Orchestrator

Vertical Application Orchestrator is responsible to orchestrate the cloud-native application per se in programmable resources that are provided by the infrastructure/telco-provider while the SFVO is responsible to provide the aforementioned resources along the prerequisite end-to-end connectivity. Such connectivity is achieved through programmability of the telco resources. VAO aims to help them to satisfy their vertical customers' need, by creating a fruitful environment where network-intensive services can be easily prototyped and quickly deployed into production.

In that concept VAO allows software developers to compose applications following a conventional microservices-based approach where each component can be independently orchestrated. Based on the conceptualisation of metamodels (application component and graph metamodels), they can formally declare information and requirements -in the form of descriptor- that can be exploited during the provisioning and management.

Such information and requirements may regard capabilities, envisaged functionalities and soft or hard constraints that have to be fulfilled and may be associated with an application component or virtual link interconnecting two components within an application graph. The produced application is considered as 5G-ready application.

Application developers/providers are able to go through the developed applications (published to a repository similar to a marketplace) and specify policies and configuration options for their optimal deployment and operation. Based on the provided application descriptor, application providers are able to design operational policies and formulate a slice intent. These operational policies describe how the application components should adapt their execution mode in runtime. On the other hand, the slice intent includes a set of constraints that have to be fulfilled during the placement of the application and a set of envisaged network functionalities that have to be provided. In a sense application-aware network slice intent is a declarative way from the application perspective to specify the operational goals that a network should meet and outcomes that the network is supposed to deliver. Those goals and outcomes are specifying what to accomplish not how to achieve it.

This information is used by the Vertical Application Orchestrator to request from the SFVO (or any OSS system that manages the underneath programmable infrastructure) the creation of an appropriate application-aware network slice.

To serve the particular purpose of an application graph and to fulfil the vertical application needs, the allocation of the necessary resources (both network and compute resources) with an optimized topology and the appropriate isolation are mandatory.

3.3.2.1 Vertical Application Orchestrator Architecture

The Vertical Application Orchestrator architecture is covering two layers:

- The Application Composition Layer is oriented to software developers.

- The Application Management Layer is oriented to application providers/operators.
- The Application Orchestration Layer is oriented to the deployment and the application lifecycle management.

In detail, the Application Composition Layer takes into account the design and development of applications per industry vertical, along with the specification of the associated networking requirements. The associated networking requirements per vertical industry are tightly bound together with their respective applications' graph, which defines the business functions, as well as the service qualities of the individual application.

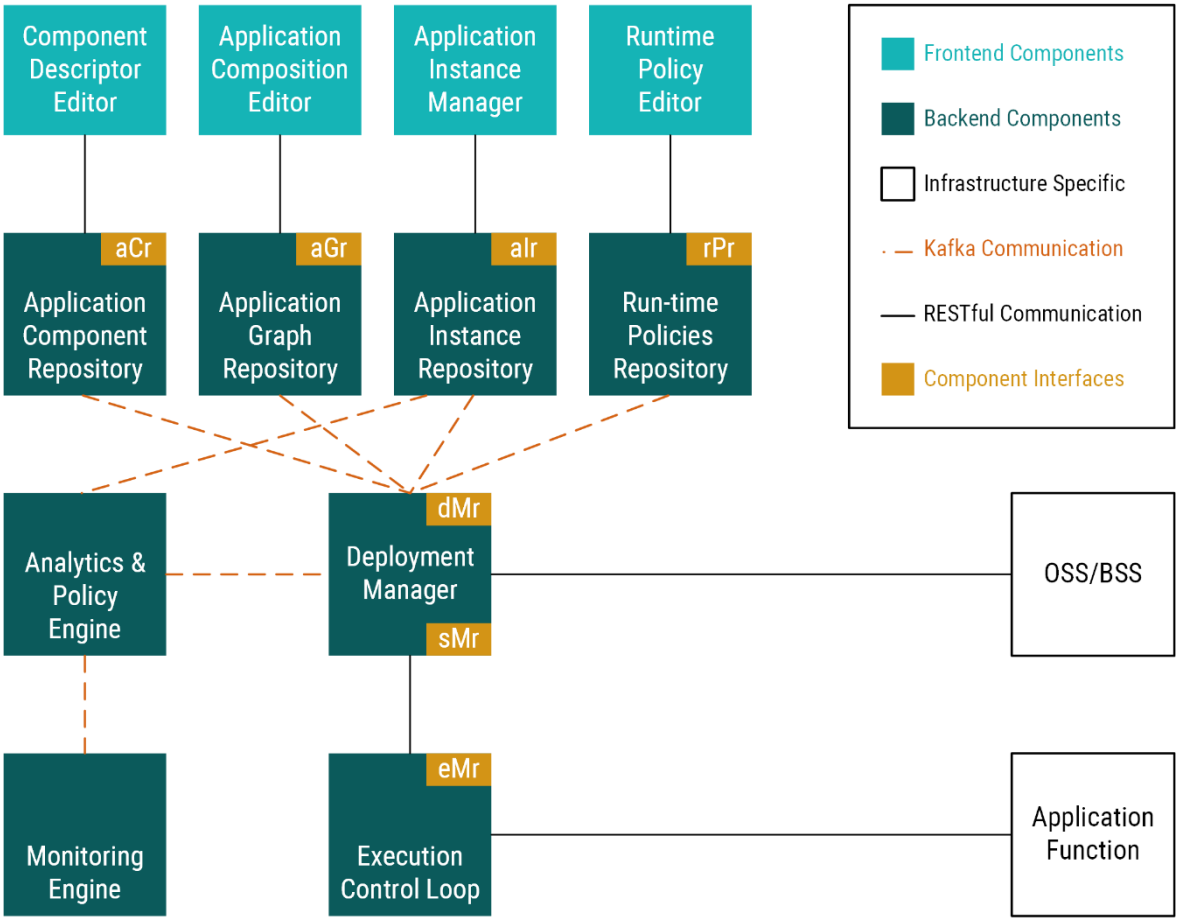

Figure 3.23: Components of the Vertical Application Orchestrator

The Application Management Layer is responsible for managing the application's operation over the application-aware network slice. Service discovery mechanisms that keeps track of the operational state of all vertical applications and their components along with monitoring streams that are generated and managed from the operation of the various components/layers. The monitoring system is responsible for the management of the metrics captured from the various infrastructure components, the management of alerts and events based on these metrics, and the visualization of the available data. Based on

the collected metrics there is an inference engine that can manipulate the operation of the application in order to fulfil and maintain customer's SLAs.

The Applications' Orchestration Layer supports the dynamic on-the-fly deployment and adaptation of the applications to its service requirements, by using a set of optimisation schemes and intelligent algorithms to provide the needed resources across the available multi-site programmable infrastructure.

VAO's components are classified as backend and front-end components. The frontend components are related to the GUI of the VAO where the application composition takes places, ie the Application Development Environment, while the backend refers to the repositories needed for the front components as well as components that have to do with the service discovery service, the monitoring, analytics and metric collection of the application components.

It is worth mentioning that VAO have been designed with 'scaling' as first-class citizen, and for that all UI-serving components are stateless.

3.3.2.2 Front End Components

Component Description Editor: A graphical form for onboarding and declaring requirements and constraints per application component.

Application Composition Editor: A graphical editor for composing an application graph by selecting the appropriate components and connecting them.

Application Instant Manager: A graphical interface for collecting and showcasing various information about the different application instances.

Runtime policy editor: Runtime policy definition is realised by the end users or clients and is applicable to an application graph. Each runtime policy consists of a set of rules (or expressions) that are examined during runtime. This editor provides a way for the user to create runtime policies.

3.3.2.3 Back End Components

Application Graph Repository: A repository for storing all the application graphs. As already described, an application graph is practically a directed acyclic graph (DAG) of application components. Practically an application graph is a serialization as DAG.

Application Instance Repository: An Application instance is the entity that represents a parameterized Application that will be deployed through SFVO in a programmable infrastructure.

Runtime Policy Repository: A repository for storing user's policies.

Component Repository: A repository for storing all the onboarded components through the VAO.

Monitoring, Analytics and Policy Engines: The monitoring with the analytics and policy components are considered tightly coupled. The Analytics engineconsumes and analyses information from the monitoring engine to identify trends and meaningful patterns. The Policy Engine provides policies enforcement over the deployed applications following a continuous match-resolve-act approach. Specifically, the match phase regards the mapping of the set of applied rules that are satisfied based on the alerts coming from the monitoring engine. Policy enforcement is realised through a rule-based framework that attempts to derive execution instructions based on the current set of data and the active rules; rules associated with the deployed application at each point of time. The Policy engine provides the suggested actions on the Deployment Manager and the Execution Loop to materialize these actions.

OSS API: This interface (is also mentioned as sub-part of the Deployment Manager) is responsible for generating the Slice Intents that are required in order for an application instance to be operational. The Slice Intent is all of the requirements and the constraints that the application requires in order to operate smoothly. The slice intent is handed over to any OSS system responsible for the programmable infrastructure and the for the allocation of the required resources.

Deployment Manager: The deployment manager is responsible to handle the deployment and un-deployment of an Application Graph over the programmable infrastructure. The Deployment manager is responsible to monitor the instantiation of the vertical components within the deployment unit(VM, Pod etc). This is practically performed by a component that is addressed as Agent and is loaded in each VM/Pod that is spawned. The Agent is responsible to report on the success boot sequence of the vertical components and even react on managed exceptions (e.g., VM is not available at the moment, component is loaded but health check is failing).

Deployment Manager identifies the following acceptable states of a deployment process. From a single component point of view, the deployment status can be:

- BOOTSTRAPPED
- ONGOING
- DEPLOYED
- STARTED
- FAILED
- UNDEPLOY

Deployment Manager is comprised from sub-components like the following:

Agent: The agent is responsible to monitor the boot sequence of the vertical application.

Service Discovery Server (SDS): The server keeps track of the operational state of all vertical applications and their components. It acts as a cache memory.

Execution Manager Control loop: Each orchestration engine spawns one 'infinite' control loop per deployed application. This control loop is responsible for tracing the execution of all components and react on possible errors.

Elasticity Controller: It is responsible to handle the scale-in/out operations.

Slice Parser: This component is responsible to parse the response of the OSS underlying system and infer the proper deployment sites that should be used in order for deployment to take place. This component also infers the related commands that needs to be executed in order to trigger the deployment.

3.3.3 Monitoring

As extensively mentioned before orchestration level APIs are an important tool to provide uniform portal and management interfaces to end customers. Good orchestration can accelerate service onboarding, automate full life-cycle management, enhance customer experience, transform seamlessly from VNF to CNF, and simplify interoperation between telco and public cloud.
In this orchestration concept and in order to enable optimized management of application and network services it is evident the role of a monitoring mechanism. A monitoring mechanism that can provide feedback and collect the metrics across all the involved layers of the architecture. The quintessential role of such mechanism can unlock an end-to-end view of physical and virtual resources for optimization of resource management, inputs to assist security policy frameworks and decision-making systems, SLA assurance, and fault management decisions.

In FUDGE-5G one of the major challenges is the unification of the monitoring streams that are generated from the operation of the various components/layers. These metrics are categorized in the following groups:

- **Infrastructure Layer** Such metrics quantify the quality of the provided IaaS resources (from the telco-provider) during the slice creation. They refer to CPU speed, amount of memory, storage speed (IOs per second), etc.
- **Platform Layer** These are metrics that are measured within the administrative domain of the OSS/Telecommunication Provider and they are performed by vendor-specific software.
- **Enterprise Services Layer** Such metrics quantify the several execution parameters that can be measured passively, i.e. through a probe. In this category of metrics lays also the metrics that quantify the several execution parameters that are exposed by the enterprise service per se. In general, to obtain these exportable metrics some guidelines (libraries) are provided to assist the developers.

Up till now the platform layer metrics were difficult to acquire. But the ongoing softwarisation process in telecommunications and the full separation of all network services and functions from the underlying physical infrastructures enables a unified way to acquire metrics for this specific domain.

Leveraging upon this softwarisation process it is easily inferred that the unification process of collecting data streams is logical and not physical since technically is feasible. However, this unification process is still on pause due to the reluctancy of telco providers to expose specific types of monitoring streams outside of their administrative boundaries.

In FUDGE-5G the monitoring solution addresses not only on specific domain (e.g. vertical) but also on multi-site network infrastructure deployments and performs metrics acquisition from a variety of domains. Specifically, the resources to be monitored fall in one of the domains Vertical/Application oriented metrics and cross domain metrics.

3.3.3.1 Vertical Application Monitoring

Application monitoring measuring is not only including performance, availability, and user experience metrics of application software but also other metrics that align with the business objectives. In that sense, besides the more “generic” metrics (regarding resources and network usage of application component), it is important to collect application specific metrics. In the case of vertical application monitoring, the collection of custom metrics is available by providing configuration files (official name is exporters) that declares various information about what is measured and where are these metrics are exposed.

3.3.3.2 Cross Domain Monitoring

Cross Domain Monitoring includes metrics coming from:

- NFV Infrastructure (NFVI) resources that comprise of physical and virtual compute, network and storage resources
- SDN-enabled elements, including physical and virtual resources
- Physical devices that do not belong to the previous categories, such as non-SDN compliant network routers and switches for which we want to capture monitoring information
- Linux containers deployed to run application components that form the 5G-ready vertical applications.

The monitoring engine is responsible for the metrics acquisition coming from the various domains and the management of them. Based on the metrics, the monitoring engine is also responsible for activating alerts and events that triggers policies and inferences on the respective module.

3.3.3.3 Analytics and Policy Engines

As already mentioned, the Analytics and policy components are highly dependent with the Monitoring Engine. Each microservice that is running contains a Service Discovery agent that is responsible to announce the existence of the microservice and in parallel to conduct health checks. The measurements' collection system is responsible to **pull** periodically all measurements that are exposed by the measurement agent. Collected measurements are persisted based on a configurable retention period. Every specific interval an "Insights Examiner" examines various measurement datasets and in case a condition is satisfied, it notifies the Rule engine.

The Rule Engine is going to consume the provided alerts and set of monitoring streams by the Monitoring Engine. The alerts regard notifications provided upon the satisfaction of specific conditions. The monitoring streams regard streaming data for compound/aggregated metrics that are coming from the Monitoring Engine, based on processing of raw collected data.

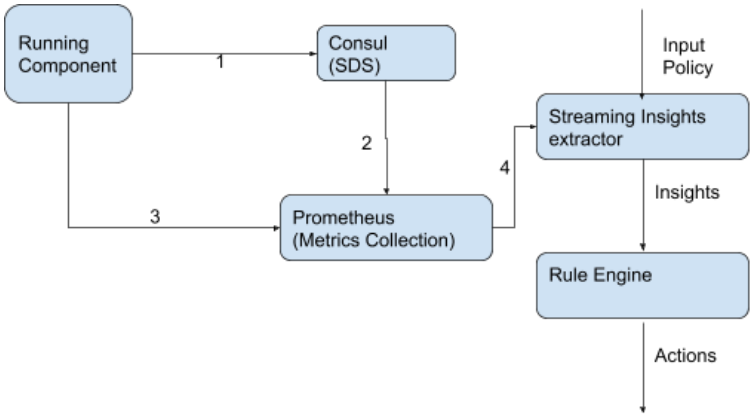

Figure 3.24: VAO policy engine

There is a need to have Policies working on a streaming mode in order to (i) support sliding time windows in the rules evaluation process, (ii) support validation of the enforced actions taking into account the time period that they were triggered, as well as (iii) avoid triggering of rules that can lead to oscillation effects (e.g. do not de-provision upon scaling, etc.). Such functionalities are considered as CEP-oriented.

3.4 End-to-End Service Slicing

Slicing in a telecommunication system has two key objectives: isolation of communication between two or more entities and the enforcement of requested QoS values for the communication. In a mobile telecommunication system the system is composed of the following sub-systems that are operating autonomous to each other when it comes to slicing but offer interfaces for configuration and information sharing purposes by other sub-systems:

- Radio Access Network

- User Plane from UEs to UEs or UEs to DNs
- 5GC Network functions
- Data Network

Key for a fully programmable and flexible E2E system slicing approach is the ability to create, modify or delete a slice for a single communication transaction (e.g. HTTP request/response) only, if deemed necessary.

3.4.1 Slicing to Enable PNI-NPNs (All 5GC Vendors, Kashif, Except O2M)

Network slicing is widely recognized as a key feature of 5G networks, as it will be used especially to address Industry 4.0 and Internet of Things (IoT) wireless requirements by provisioning operational isolation of the network, thus supporting multiple vertical customers at once. At the same time, many enterprises have been deploying business-critical Private LTE networks and experimenting with Private 5G. Indeed, the deployment of private networks is becoming a more and more common choice for private companies and public institutions that want to maximize the control capability on their own private telecommunication systems. Such NPNs fill the performance and service gaps left unaddressed by nationwide Public Land Mobile Networks (PLMNs). Interoperability among local NPNs and between each NPN and PLMNs has become therefore a major focus point for all the parties involved in the 5G conception and deployment. ¡

3.4.2 Communication Isolation of Control and User Plane Services Utilising Name-based Routing

Slicing the communication end to end focuses on resource sharing in either an opportunistic or explicit manner as well as isolating slices from each other. Furthermore, current slicing concepts in 3GPP systems merely focuses on UEs' PDU sessions and ensuring their user plane has specific QoS guarantees attached. However, in order to slice the control plane in particular in scenarios where multiple 5G Cores are deployed serving different NPNs / areas.

The current SCP realisation of NbR does not support the isolation of 5GC consumer and producer communication to ensure that NFs of one 5GC deployment cannot communicate with another one. NbR's realisation is inherited from the design of the internet where services are publicly available. However, FUDGE-5G aims to add the ability of slicing (as in isolation) 5GC deployments as an SCP feature. Thus, this section presents the necessary steps to ensure no malicious/unintended communication is being injected into the ingress point for an SCP.

To allow the isolation of service chains, the solution presented has two major categories:

1) Communicating the 5GC deployment (aka service chain) properties to the routing layer at their orchestration time
2) Run-time procedures inside the SCP to ensure isolation.

3.4.2.1 Orchestration of a Service Chain

At orchestration time of a Service Chain a resource descriptor is communicated to the SFVO which defines the service functions and their properties which includes the FQDN under which they are supposed to be reachable among other properties such as number of CPUs or the amount of memory (see Section 3.3.1.3 for more details). Once the Service Chain is orchestrated the SFVO communicates to the routing layer which Service Function Endpoint (SFE), identified through a unique identifier (e.g. its MAC or IP address), is serving which FQDN.

Figure 3.25 illustrates the message sequence chart for the orchestration and registration of an SFE against the routing layer. Note, the components illustrated in the figure entitled Service Proxy Manager, Service Proxy and Path Computation Element are NbR components that implement the SCP available to FUDGE-5G as well as the service routing on the user plane for the Media use case. More information on their functionality can be found in [FUD21].

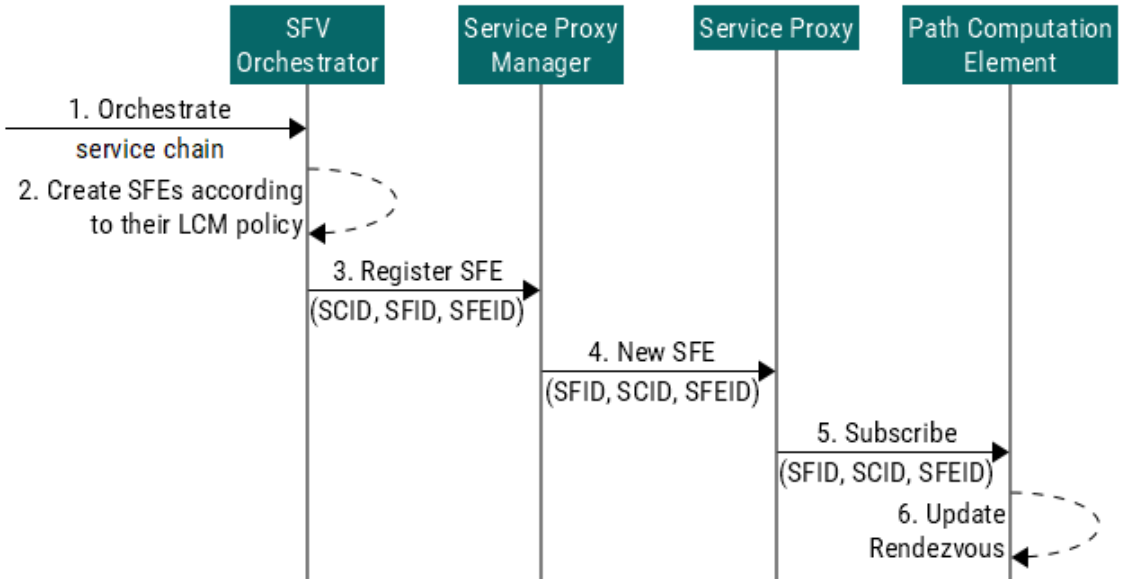

Figure 3.25: Message sequence chart for orchestrating a 5GC service chain

The steps are as follows:

1) The SFVO receives an orchestration request with a descriptor containing the service chain and its requested resources. It performs validation and interpretation tasks ensuring the validity of the provided descriptor. Part of this descriptor is a Boolean field defining whether isolation is requested or if the entire service chain can be

reached by any endpoint. The isolation toggle can be set per service chain or service function only if desired.

2) The SFVO then deploys SFEs according to their Lifecycle Management (LCM) policies.
3) Upon the successful orchestration of all SFEs, the SCO registers them against the routing layer by communicating the Service Chain Identifier (SCID), Service Function Identifier (SFID) and Service Function Endpoint Identifier (SFEID) to the SPM for each orchestrated SFE which is supposed to be exposed to the routing layer. However, as isolation is requested by the orchestrator, the SFE registration also includes the SCID which is the parent domain under which all SFs of the service chain have received unique sub domains. The SFID is the actual FQDN for an SF and the SFEID a unique communication identifier such as the MAC or IP address. For instance, if a service chain is composed of two SFs with SF_1's SFID set to *sf1.foo.com* and SF_2's SFID set to *sf2.foo.com*, the SCID is *foo.com*. An example of a json-encoded registration for SF_2 is provided in the code block below this enumerated list.
4) The SPM communicates to all SPs the list of SFEs they are now serving including the information about SCID, SFID and SFEID.
5) As a result, each SP issues the necessary subscriptions towards the PCE for the HTTP Content Identifier (CID) */http/<SFID>* with *<SFID>* being the SFID provided in the orchestration template, i.e. an FQDN. In addition to the SFID, the SCID and SFEID is also communicated to the PCE as part of this subscription request.
6) The PCE adds the subscription under the ICN root scope /http with the information item SFID (FQDN) resulting in */http/<SFID>*. In addition to the HTTP namespace, the PCE creates a second namespace that is linked with the just created CID */http/<SFID>* of format */isolation/<SCID>/<SFEID>*, as illustrated in Figure 3.26.

```
{
       "sfid": "sf2.foo.com",
       "sfeid": "00:0c:29:14:e4:5b",
       "isolate": true,
       "scid": "foo.com"
}
```

The key in the newly introduced namespace /isolation inside the rendezvous functionality of the PCE is its logically linked relationship of the information item in */http/<SFID>* with the scope */isolation/<SCID>*.

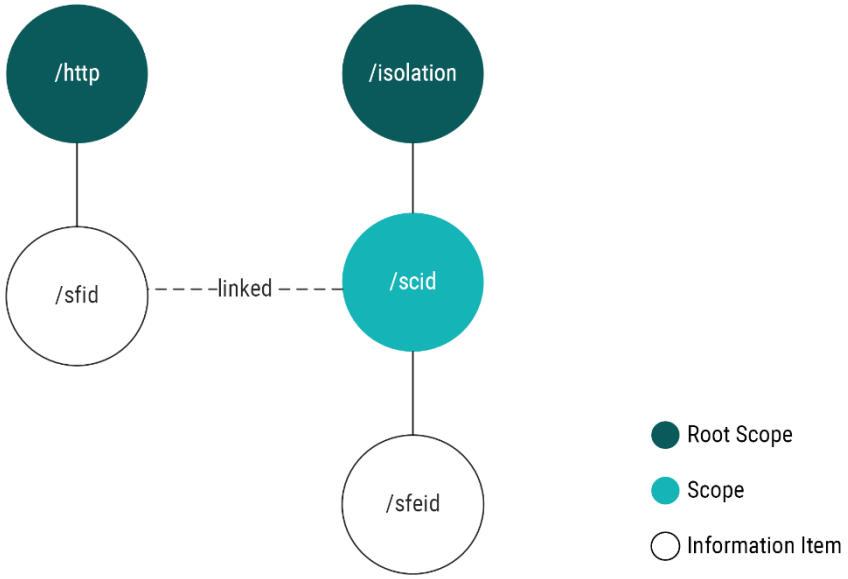

Figure 3.26: Linked HTTP and SCID namespaces inside PCE

The usage of the linked namespaces comes into play at run time and is described in the next section.

For non-HTTP traffic, the IP namespace is being used instead of the HTTP one and the SFID becomes the IP address of the server instead of the FQDN. The actual steps described above do not change except the linked namespace relationship. If non-HTTP traffic is traversing the SPs and isolation, the namespace linkage applies, as depicted in Figure 3.27.

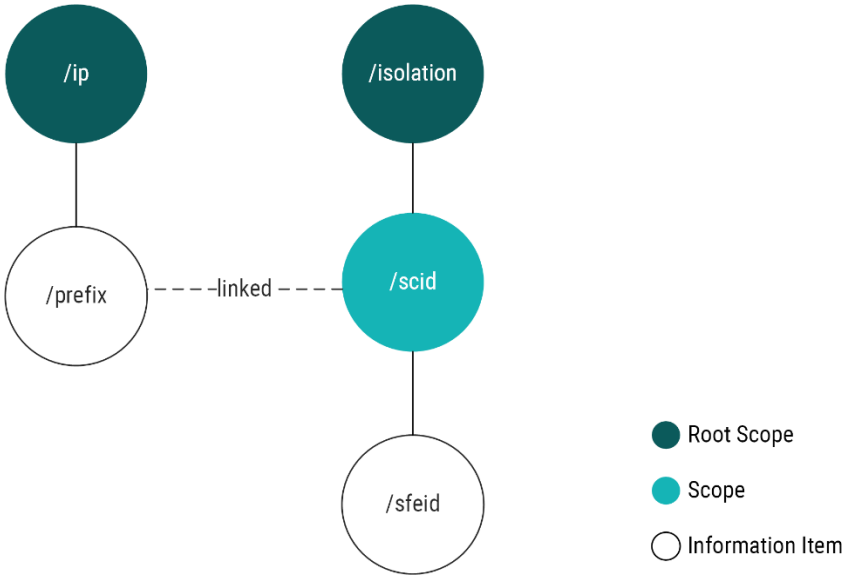

Figure 3.27: Linked IP and SCID namespaces inside PCE

3.4.2.2 Run-Time Procedures

Figure 3.28 illustrates the steps a consumer SFE, SFE_C, to communicate with a producer SFE, SFE_P, over an NbR-based SCP using the stateless protocol HTTP.

The MSC illustrates a single client and server which have their own SP, SP_C and SP_P, that are implementing the service routing of NbR. For the sake of simplicity, there is no dedicated switching fabric illustrated and both SPs can communicate directly with each other. The starting point for the steps in the MSC below is that the producer SFE, SFE_P, has been orchestrated and registered under the SFID *sfe-s.foo.com* with the SCID *foo.com*.

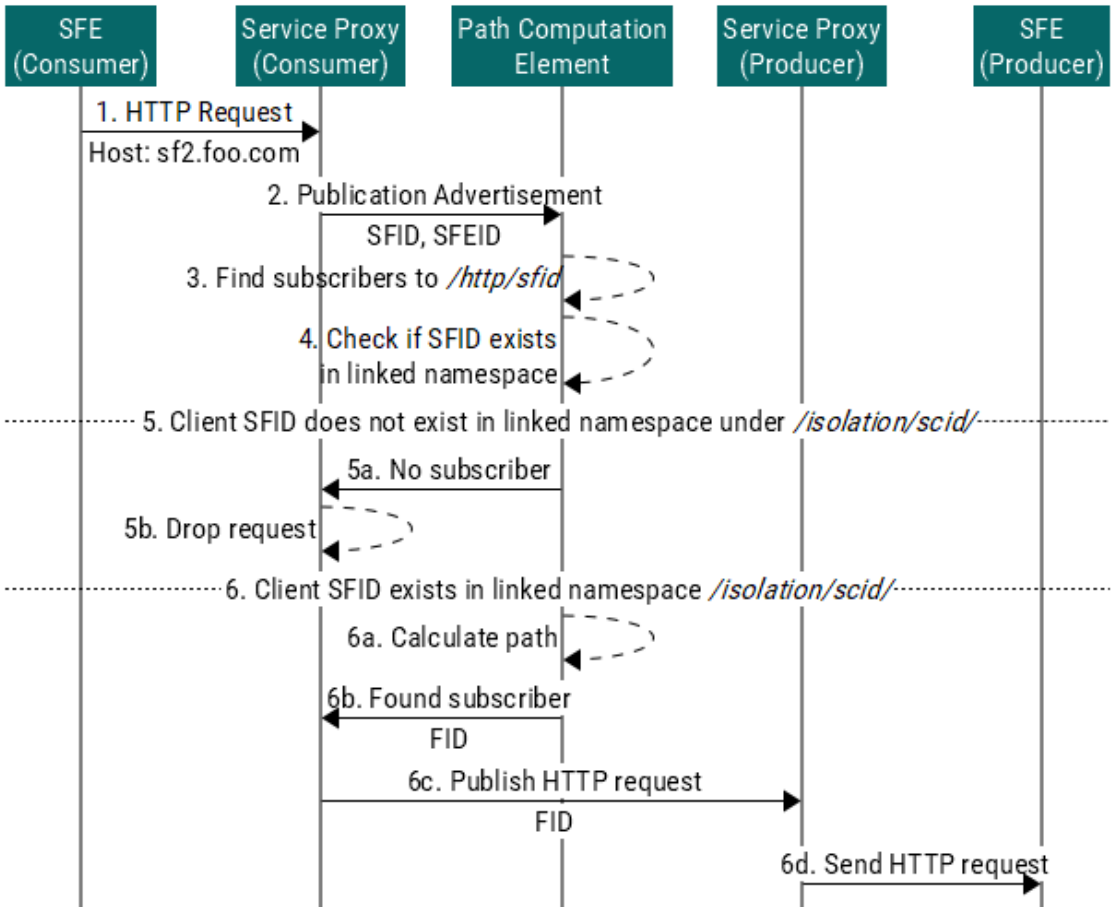

Figure 3.28: Message sequence chart for run time operations

The steps illustrated in the figure are as follows:

1) The SFE_C sends a HTTP request to *sf2.foo.com* which is received by SP_C, the SP serving SFE_C.
2) The SP_C asks the PCE for a Forwarding Identifer (FID) to the subscriber where the request is supposed to be sent to by issuing a publication advisement which includes the SFID *sf2.foo.com* as well as the SFEID of SFE_C, i.e. MAC or IP address.
3) The PCE first checks the HTTP namespace for potential subscribers. If there is no subscriber, the PCE immediately communicates that to the SPC which will drop the HTTP request. For simplification purposes, this is not shown in the MSC.

4) Assuming the PCE has found the communicated SFID in the HTTP namespace, the PCE now checks if the SFEID is available in the linked isolation namespace which points to */isolation/foo.com*.
5) This step describes the case when the PCE does not find the SFEID in the linked isolation namespace:
    5a. The PCE cannot find the SFEID in the isolation namespace and therefore does not allow the client SFE to reach the SFE server. The PCE informs the client SP SP_C that no subscriber is available.
    5b. SP_C drops the HTTP request and ends the HTTP transaction (i.e. closing TCP or UDP socket).
6) This step describes the case when the PCE does find the SFEID in the linked isolation namespace:
    6a. As a result of finding the SFEID of the client SFE under */isolation/foo.com/*, the PCE computes the path from SP_C to SP_S.
    6b. The PCE communicates the path in form of a Forwarding Identifier (FID) to the SP_C. The FID is used for the path-based forwarding NbR implements.
    6c. The SP_C publishes the HTTP request to the SP_P.
    6d. The SP_P sends the HTTP request to the SFE_P.

3.4.3 End-to-End Slicing Orchestration

The slicing orchestrator is in charge of the guarantee of the negotiated SLA of the slice in an end-to-end path. Most of the time, this capability requires undertaking multiple SLA negotiation in parallel with different technical domains (e.g., the RAN controller, the SCP, and the VAO), and to guarantee consistence and coherence in the created slices via proper configuration of the underlying domains.

The Service Level Management (SLM) is the process of negotiating SLA and ensuring that these are met. It is one key element for planning the slice from the perspective of linking technical domain capabilities to the operation demand of all stakeholders. Thus, the SLM framework defines the interaction between business and network operations process. It considers the following activities:

- Planning: slice request and SLA definition
- Deployment: SLA negotiation and technical slice requirements definition
- Operation: Usage and technical slice requirements reporting

The architecture of the orchestrator separates the slice requests from their actual delivery so that any local configuration and operation on a technical domain is transparent from the point of view of the slice consumer, remaining at the same time compliant to its business needs.

In this perspective, the lifecycle management, a fundamental capability of SLM, is split into 2, with slice–specific and domain-specific lifecycle management. While the slice-specific lifecycle management is the sole responsibility of the slicing orchestrator, domain-specific management is delegated to the domain controller. This segregation leverages on the abstraction hierarchy for describing service, network configuration and device configuration using different data models. One of the advantages of this approach is also that it allows backward compatible end-to-end slicing orchestration, by simply adapting the slicing capabilities to the means offered by the technical domains. On the downsides, a single component could potentially hinder achieving fine grained control of the end-to-end slice.

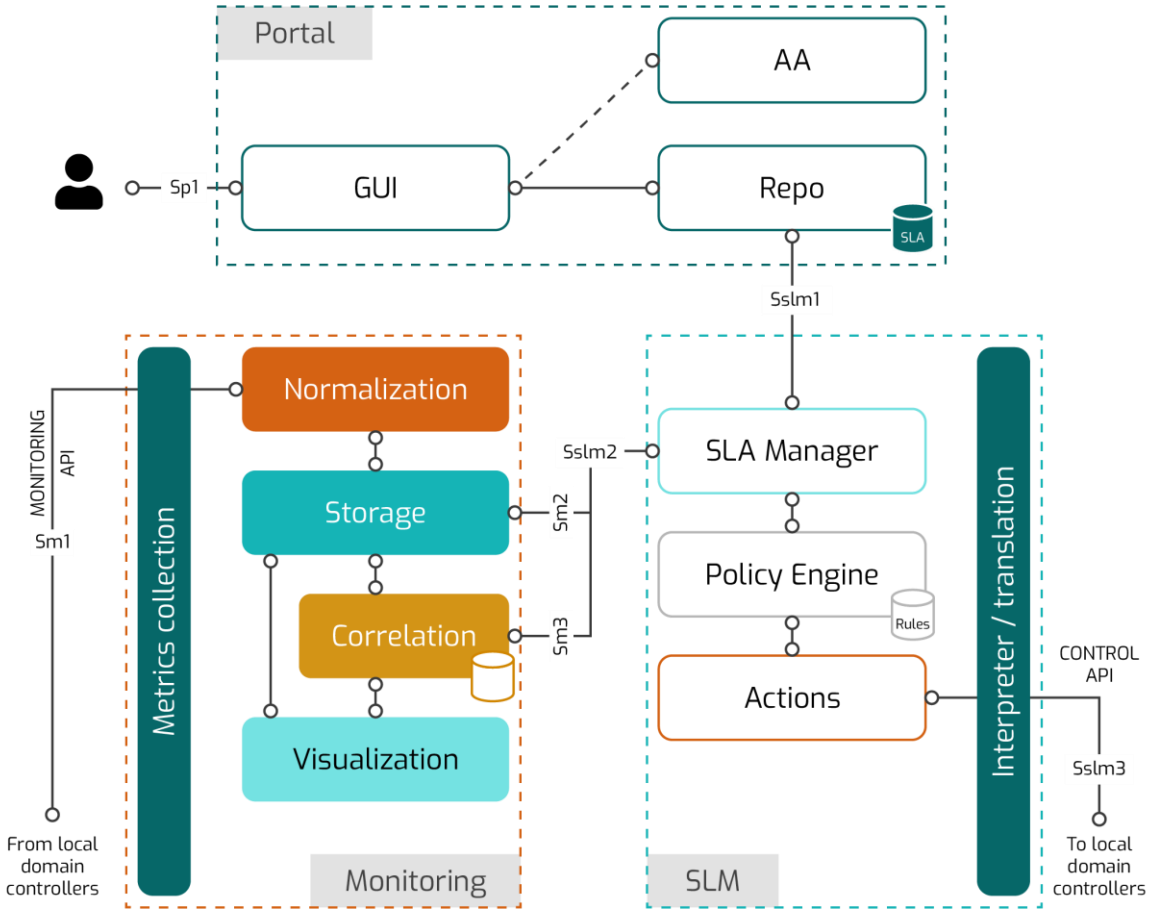

Figure 3.29: Components of the end-to-end slice orchestrator

As per Figure 3.29, the end-to-end slice orchestrator architecture is composed of three tightly-connected high-level components. Their roles are explained in the following.

Portal

The main objective of the portal component is to provide the SLM block with a slice model. It is in charge of the interaction with the end user via a GUI, proposing both graphical and textual model, allowing the user first to define the slice model and its associated SLA, then to on-board them to the orchestrator. This component hosts as well an authentication and authorization component that checks user permissions, and the repository that stores the already created slice templates.

The **Sp1** interface allows the end user to interact with the Portal, via a HTTPs interface connecting with a web server hosting the GUI.

Monitoring

The main objective of the monitoring block is to provide the SLM block with a set of live KPIs for the slices in operation. It is in charge of pulling/receiving the telemetry data from the deployed technical domains. It is composed of a powerful data analysis pipeline that allows to normalize, enrich, and transform data, store both raw and processed data for later processing and forensic analysis, correlate together multiple source of data, performing as well complex processing on them, and provide a dashboard for user visualization.

The **Sm1** connects the data sources (e.g., the local domain controllers) to the ingestion of the data at the monitoring system via specific monitoring APIs. In order to be interoperable with a large number of implementation, this interface should implement a broad set of ready to use input-output connectors, including ELK Beats, Prometheus exporters, Kafka, Netflow, SNMP, Lumberjack, and others.

The **Sm2** and the **Sm3** interfaces allow the communication between the monitoring component (respectively the storage and the correlation blocks) and the SLM component, to exchange the processed events providing the KPIs for the slices. Typically, this interface implements a streaming engine compliant with a dataflow processing model such as Spark, Kafka, or Flink.

Service Level Manager (SLM)

The SLM component is in charge of managing the slice during its operation and configuring the local domain controllers. The SLM interfaces both with the monitoring component to retrieve situation awareness, and with the slice repository, providing the slice model together with the expected SLA. It is thus possible for the SLA manager to compare the retrieved KPIs from the slices currently in operation against the SLA requirements. In case of non-compliance, or for optimization purposes, a Policy engine engages into selecting and tuning a set of predefined rules. These are translated into actions that have to be communicated to the list of local controllers through a set of control APIs.

The **Sslm1** interface offers the possibility to exchange the slice and the SLA model between the portal and the SLM component. In general, this interface will implement an API REST.

The **Sslm2** interface is the receiving side of the **Sm2** and the **Sm3** interfaces.

The **Sslm3** interface allows the SLM component to communicate directly with multiple local domain controllers via a set of control APIs. This interface should implement the largest possible selection of protocols, in order to be able to communicate with a multitude of domain controllers. In particular, it should be capable of communicate via REST and websocket based APIs.

4 Conclusion

This section concludes the interim release of the FUDGE-5G architecture including its components and interfaces. With its rather unique proposition to define the FUDGE-5G platform as an evolved SBA platform that offers the functionality service routing, orchestration, telemetry and slicing as unified platform services, it allows to demonstrate how the realisation of the “cloud-meets-telecom” slogan can positively affect the dynamicity, elastically and flexibility requirements for NPN deployments and the opportunities within.

The deliverable presented the current state of the art in the area of Service-based Architecture, 5GLAN, Time Sensitive Networking, 5G Multicast and end-to-end slicing and their the relationship to FUDGE-5G. Also, the current affairs in OAM procedures of how 5GCs are provisioned and configured is described demonstrating the scattered landscape of approaches and technologies demonstrating a clear need for an evolved Service-based Architecture that further cherishes and adopts cloud principles. This formed the foundation for the proposed eSBA system architecture in this document that is composed of routing, orchestration, telemetry and slicing capabilities offered to enterprise services which are 5G Cores and vertical applications. Furthermore, the eSBA proposition applies equally to control as well as user planes and has been designed for that purpose in mind.

The work presented in this deliverable is an intermediate release of the FUDGE-5G system architecture and will see further details, clarifications, corrections and additions where discovered in the final FUDGE-5G system architecture deliverable D1.3.

5 References

[3GP16] 3GPP, "TR 23.799: Study on Architecture for Next Generation System", Release 15, 2016. Online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3008

[3GP18] 3GPP, "TR 23.501: System architecture for the 5G System", Release 15, 2018. Online: https://www.3gpp.org/ftp//Specs/archive/23_series/23.501/23501-f10.zip

[3GP19a] 3GPP, "TR 22.261, Service requirements for next generation new services and markets", 2019. Online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3107

[3GP19b] 3GPP, "TR 23.501: System architecture for the 5G System", Release 16, 2019. Online: https://www.3gpp.org/ftp//Specs/archive/23_series/23.501/23501-g10.zip

[3GP21] 3GPP, "TR 23.247: Architectural enhancements for 5G multicast-broadcast services", Release 17, 2021. Online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3854

[BAR20] David Gomez-Barquero, Jordi J. Gimenez, and Roland Beutler, "3GPP Enhancements for Television Services: LTE-Based 5G Terrestrial Broadcast", Wiley Encyclopedia of Electrical and Electronics Engineering, 2020, DOI: 10.1002/047134608X.W8410

[BAR20a] Barakabitze, A. A., Ahmad, A., Mijumbi, R., & Hines, A., "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges", Computer Networks, 2020. DOI: 10.1016/j.comnet.2019.106984

[BLO20] Marcel Blöcher; Ramin Khalili; Lin Wang; Patrick Eugster, "Letting off STEAM: Distributed Runtime Traffic Scheduling for Service Function Chaining", IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, DOI: 10.1109/INFOCOM41043.2020.9155404, 2020.

[CAL15] Jordi Calabuig, José F. Monserrat, and David Gómez-Barquero, "Broadcasting in 4G mobile broadband networks and its evolution towards 5G", PhD thesis, 2015. Online: https://riunet.upv.es/bitstream/handle/10251/48561/CALABUIG%20-%20Broadcasting%20in%204G%20mobile%20broadband%20networks%20and%20its%20evolution%20towards%205G.pdf?sequence=1

[COR19] Coronado, E., Khan, S. N., and Riggio, R, "5G-EmPOWER: A software-defined networking platform for 5G radio access networks", IEEE Transactions on Network and Service Management, 16(2), 715-728, 2019, DOI: 10.1109/TNSM.2019.2908675

[DET19] Detecon Consulting, "5G Campus Networks - An Industry Survey", 2019. Online: https://www.detecon.com/en/about-us/news/industry-survey-5g-campus-networks

[DRA17]	Draxler, H. Karl, M. Peuster, H.R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, G. Xilouris, "SONATA: Service programming and orchestration for virtualized software networks", Proceedings of the IEEE International Conference on Communications Workshops, 2017, DOI: 10.1109/ICCW.2017.7962785.
[ENG18]	Princen Alice, "Twelve-Factor Apps, Sidecars, and Big Data", 2018. Online: https://engineering.wheresalice.info/2018-03-04/12factor-app-big-data/
[ETS17]	ETSI Industry Specification Group, "GR NFV-EVE 012 Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework", 2017. Online: https://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/012/03.01.01_60/gr_nfv-eve012v030101p.pdf
[ETS20]	ETSI Industry Specification Group. "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Requirements for service interfaces and object model for OS container management and orchestration specification", 2020. Online: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/040/04.01.01_60/gs_NFV-IFA040v040101p.pdf
[FUD21]	The FUDGE-5G consortium, "Deliverable D2.1: FUDGE-5G Technology Components and Platform – Interim Release", 2021. Online: https://www.fudge-5g.eu/en/deliverables
[GSM17]	GSMA, "An introduction to network slicing", 2017. Online: https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf
[HAR09]	Frank Hartung, Uwe Horn, Jörg Huschke, Markus Kampmann, and Thorsten Lohmar, "MBMS—IP Multicast/Broadcast in 3G Networks", Convergence of Digital TV Systems and Services, 2009. DOI: 10.1155/2009/597848
[IEE02]	Institute of Electrical and Electronics Engineers, Inc., "IEEE 1588-2002 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2002. Online: https://standards.ieee.org/standard/1588-2002.html
[IEE08]	Institute of Electrical and Electronics Engineers, Inc., "IEEE 1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008. Online: https://standards.ieee.org/standard/1588-2008.html
[IEE21]	Institute of Electrical and Electronics Engineers, Inc., "Time-Sensitive Networking Task Group", 2021. Online: https://www.ieee802.org/1/pages/tsn.html
[ITI19]	ITIL Foundation, "Business Practises for IT Service Management". 2019. Online: https://www.officialitil4.com/
[ITU12]	ITU-T, "Y.3011, Framework of Network Virtualization for Future Networks", 2012. Online: https://www.itu.int/rec/T-REC-Y.3011-201201-I
[KAT16]	Katsalis, K., Nikaein, N., Schiller, E., Favraud, R., & Braun, T. I., "5G architectural design patterns", In 2016 IEEE International Conference on Communications Workshops (ICC) (pp. 32-37), DOI: 10.1109/ICCW.2016.7503760, 2016.

[MIC17] Microsemi, "Profile for Use of Precision Time Protocol in Power Systems", Online: https://www.microsemi.com/document-portal/doc_view/133171-precise-synchronization-for-electrical-utilities-the-ptp-power-profile

[NGM15] NGMN, "5G Whitepaper", 2015. Online: https://www.ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf

[RES18] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, 2018. Online: https://www.rfc-editor.org/info/rfc8446

[ROO19] S. Roomer, P. Hedman, M. Olsson, L. Frid, S. Sultana and C. Mulligan, "5G Core Networks, Powering Digitalization", Elsevier, 2019.

[RIC15] M. Richards, "The challenges of Service-Based-Architecture," Oct 2015. Online: https://nofluffjuststuff.com/magazine/2015/10/the_challenges_of_service_based_architecture

[SAV16] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch and R. White, "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)", RFC7868, Online: https://datatracker.ietf.org/doc/html/rfc7868

[TEE08] Michael D. Johas Teener and Geoffrey M. Garner, "Overview and timing performance of IEEE 802.1AS", IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, 2008, DOI: 10.1109/ISPCS.2008.4659212

[TEL19] Deutsche Telekom, "5G Technology in Campus Networks," Nov 2019. Online: https://www.telekom.com/en/company/details/5g-technology-in-campus-networks-556692

[TRO19] Trossen, D. and Robitzsch, S. and Hergenhan, S. and Riihijarvi, J. and Reed, M. and Al-Naday, M., "Service-based Routing at the Edge", arXiv preprint arXiv:1907.01293, 2019, http://arxiv.org/abs/1907.01293