Chapter 7

Towards Continuously Programmable Networks

By Dimitris Tsolkas, et al.[1]

DOI: 10.1561/9781638282396.ch7

Published in *Towards Sustainable and Trustworthy 6G: Challenges, Enablers, and Architectural Design* by Ömer Bulakçı, Xi Li, Marco Gramaglia, Anastasius Gavras, Mikko Uusitalo, Patrik Rugeland and Mauro Boldi (eds.). 2023. ISBN 978-1-63828-238-9. E-ISBN 978-1-63828-239-6.

Suggested citation: Dimitris Tsolkas, *et al.* 2023. "Towards Continuously Programmable Networks" in *Towards Sustainable and Trustworthy 6G: Challenges, Enablers, and Architectural Design*. Edited by Ömer Bulakçı, Xi Li, Marco Gramaglia, Anastasius Gavras, Mikko Uusitalo, Patrik Rugeland and Mauro Boldi. pp. 270–321. Now Publishers. DOI: 10.1561/9781638282396.ch7.

now
the essence of knowledge

1. The full list of chapter authors is provided in the Contributing Authors section of the book.

7.1 Introduction

While programmability has been a feature of network devices for a long time, the past decade has seen significant enhancement of programming capability for network functions and nodes, spearheaded by the ongoing trend towards softwarization and cloudification. In his context, new design principles and technology enablers are introduced (Section 7.2) which reside[2] at: (i) service/application provisioning level, (ii) network and resource management level, as well as (iii) network deployment and connectivity level.

At the service/application provisioning level, the exposure of Application Programming Interfaces (APIs) from the network core and edge creates new opportunities to third parties for interaction with the network [1]. The work of 3rd Generation Partnership Project (3GPP) SA6 towards vertical application enablers

2. The categorization used here is from a programmability and enabler point of view and is not strictly mapped to the Chapter 2 architectural structure.

(VAEs) is a representative paradigm in this field. At the network and resource management level, there are disruptive changes at all the domains of the service provisioning chain. Key enablers can be considered the adoption of the cloud-native approach from the communication service providers (CSPs), as well as the recent work from Open RAN Alliance (ORAN) towards vendor-agnostic management of radio access components. Finally, at the network deployment and connectivity level (including edge compute capabilities exposed to third parties), the deployment of private networks is well specified already, while the concepts of the Cell-Free paradigm and the provisioning of a connectivity mesh topology to end devices are expected to support the vision of a truly flexible access network.

In this context, disruptive architectural and service concepts emerge, anticipating multi-connectivity structures (multiple coordinated point-to-point connections), potentially including edge compute and third-party service function chains. In addition, service abstractions, programmability features, and new application with capabilities to interact and negotiate with the network, are being developed accordingly. The notion of third party is re-contextualised, including two main views: (i) one is within the 6th Generation (6G) system (platform), for network applications that are developed by third parties and (ii) the other is on top of the 6G system (platform), for services provided from third parties through the system (platform) and might be specific to a vertical.

Due to the above-mentioned multilevel evolution, a total transformation of the conventional mobile networks is expected towards operating as open service provisioning platforms. The cornerstone of this transformation is the definition of common interfaces and reference points that enable interaction by third parties with the network functions and nodes at any of the above-mentioned levels and for all the communication planes (control, management, and data plane).

The realization of such interaction is facilitated through various types of interaction-enabling frameworks (representative frameworks can be found in Sections 7.3, 7.4, 7.5, and 7.6) that on their southbound can securely consume native APIs and access network nodes (e.g., switches, gNBs, and NFs of network core) to on-board new applications or enforcing new policies, while on their northbound they can expose (e.g., vertical-oriented) services to support any kind of network-aware application and services (see Figure 7.1). Those frameworks shall take advantage of programming languages, such as the protocol-independent packet processors – P4 [2] (further explained in Section 7.5), as well as common data models, such as the OPC UA[3] information models which refer to vertical-specific companion specifications for the industrial/manufacturing nodes.

3. https://opcfoundation.org/developer-tools/specifications-opc-ua-information-models

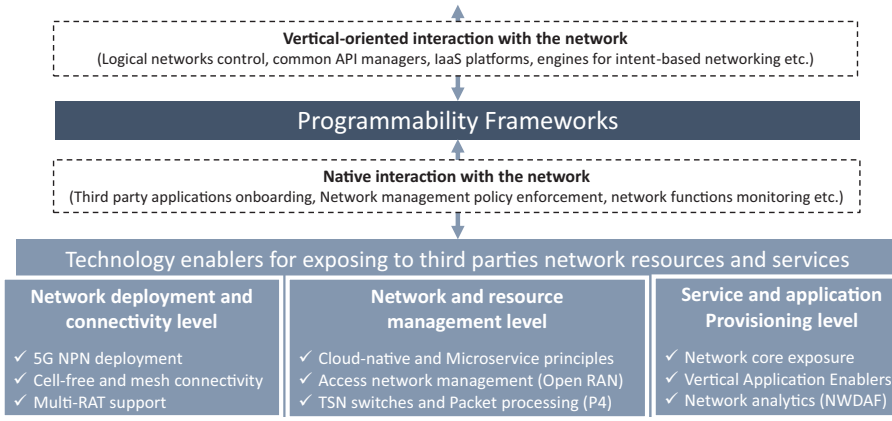

Figure 7.1. Enabling interaction of verticals with the underlay network.

Overall, the implementation of frameworks that exploit common/standardized languages, APIs, and data models provides the means for the enforcement of programmability in the next-generation networks, a concept that incorporates the capability of a device or a network to accept a new set of instructions that may alter the device or network behaviour [3]. A representative example of the programmability potential is the concept of intent-based network (IBN) which has emerged to introduce a layer of artificial intelligence (AI) in the 6G networks. It promises to solve problems of traditional networks in terms of efficiency, flexibility, and security [4, 5]. This technology has revolutionized the way that interaction is performed with systems by starting to communicate through intents. This paradigm is further studied in Section 7.7.

From the business perspective, the above-mentioned programmability frameworks create a new potential around the development of the so-called network applications. Network applications (or Network Apps) are third-party applications that interact (through standardized APIs) with the network to provide network- or vertical-oriented services. Network applications provide network- or vertical- oriented services, meaning that they can assist/enhance either the network operation/management[4] or the vertical application.[5] As third-party apps, the Network Apps should interact with network functions/nodes though open and

4. For instance, in the EVOLVED-5G project, a related contribution to 3GPP SA6 work has emerged (3GPP/TSG SA2/eNA_Ph2 Rel.17: Contribution "Support of DN performance analytics by NWDAF" – S2-2101388) under the scope of extending the NWDAF analytics APIs so that network applications can retrieve data from vertical apps, and the NWDAF build performance analytics and predictions by using inputs from network applications.

5. The ICT-41 projects (5GPPP, phase 3, part 6 projects), work towards providing network applications that fulfil needs and requests from various vertical industries, e.g., automotive (5GIANA, 5GASP), Industry 4.0/manufacturing (5GINDUCE, EVOLVED5G, 5GERA), transport & logistics (VITAL5G, 5GERA),

standardized interfaces/APIs that can reside at any plane (user, control, management) or any domain (core, radio, transport). To support the Network Apps lifecycle, experimentation facilities emerge, which share common principles and functionalities as summarized in Section 7.8 (for more information, see [14]).

7.2 Technology Enablers for Network Programmability

Towards the 6G era, a continuous evolution of mobile systems is foreseen, by the introduction of new design principles and technology enablers. Considering network programmability, those enablers, could reside at: the *service/application provisioning level*, the *network and resource management level*, as well as the *network deployment and connectivity level*.

7.2.1 Enablers at Deployment and Connectivity Level

At the deployment level, programmability can be facilitated by the introduction of on-demand deployment capabilities as well as through extensibility at the access and transport domain. On the one hand, the concept of on-demand deployment refers to the effort towards creating the technologies and the business potential for establishing mobile networks in a similar way that local area networks are currently deployed. This refers to the concept of 5th Generation (5G) private networks [4]. On the other hand, extendible deployment refers to the interfacing capabilities that mobile networks provide for engaging other network technologies at the transport and access domain, such as the N3WIF – Non-3GPP Interworking Function or the Time-Sensitive Networking (TSN) gateway. All these concepts *transform the monolithic communication infrastructure of a mobile network (conventionally dedicated to only mobile network customers) to a platform that can engage different access and transport technologies (i.e., support different types of devices and protocols) and can be fully accessible by the vertical provider when it comes to private and ad hoc developments*.

7.2.1.1 Non-public networks

Already, 5G has brought the concept of the so-called 5G non-public networks (5G NPN) [4]. 5G NPN refers to a 5G system deployment that is dedicated to providing 5G network services for private use (i.e., to a specific organization, e.g., inside a factory). Such a network is deployed on the organization's premises, for instance, inside a factory. The introduction of such networks is beneficial for the verticals for a series of reasons, but, mainly due to the potential to isolate from other (public)

media (5GMediaHub), public protection and disaster relief (5G-EPICENTRE, 5GERA, 5GGASP), healthcare (5GERA).

networks and the capability to provide to the vertical users administrative and management services on the mobile network. This isolation is desirable for reasons such as performance, security, and privacy. For those reasons, the actual implementation and utilization of NPN are expected to be part of 6G networks as well. Already there are a few deployment options for the 5G NPN that can be primarily clustered into two groups.

- Standalone non-public network: an isolated network operated by an NPN operator. The NPN operator can be either the vertical itself, by using locally available 5G spectrum, or by an operator (that potentially deploys also the NPN at vertical's premises), by using licensed 5G frequencies. The key aspect in this category is that deployments do not rely on network functions provided by a public network.
- Public network integrated NPN: a non-public network deployed with the support of a public network. The above-mentioned support can include different levels of interaction with the public network at any domain of the service provisioning chain (RAN, edge, or core). The key enabler for this deployment is the concept of network slicing.

7.2.1.2 Non-3GPP interworking function

From the previous generation of mobile networks, has emerged the potential to unify heterogeneous access networks to mobile cellular technologies (e.g., [6]). 5G materializes this potential, by exploiting the concepts of untrusted non-3GPP access (introduced in Release 15), and trusted non-3GPP access (introduced in Release 16 [7]). This is realized by the addition of non-3GPP access network and wireline access support via the Non-3GPP Interworking Function (N3IWF) component, i.e., the same 5G core network (CN) is used to provide services to a wide range of wireless and wireline access technologies, enabling integration and convergence between new and legacy networks. The way that this concept will be exploited in 6G networks is to be defined; however, the combination of multiple access technologies under a common control and management system is a key feature that has not yet revealed its full potential.

7.2.1.3 Support of time-sensitive networking

Today, the vast majority of communication technologies used in manufacturing are various Ethernet-based technologies (e.g., Sercos®, PROFINET®, and EtherCAT®) [8] and field buses (e.g., PROFIBUS®, CAN®, etc.).[6] To overcome this

6. "Industrial communication technology handbook," 2nd edition, Richard Zurawski, CRC Press, August 2014.

heterogeneity, the objective of the IEEE TSN task group[7] is to provide deterministic services through IEEE 802 networks, i.e., guaranteed packet transport with bounded latency, low packet delay variation, and low packet loss. There have been defined several standards covering aspects, such as synchronization, stream reservation, pre-emption, scheduling, and *Frame Replication and Elimination for Reliability*. Thus, TSN is an important functionality of industrial communication networks. Such industrial communication networks are usually IEEE 802.1-based networks with Ethernet links (non-3GPP networks). The IEC/IEEE 60802 profile[8] specifies the application of TSN for industrial automation and describes what a mobile network (5G or beyond 5G) needs to support. The challenge in future networks is the proper integration of mobile and TSN networks as it requires the acquisition of service requirements that are not yet in the scope of the current 5G QoS framework. This is also related to the capability to achieve end-to-end URLL (ultra-reliable and low latency) communications.

7.2.2 Enablers at the Management Level

At the management level, programmability can be materialized by the cloud toolbox (which has recently supported the network slicing concept) as well as by the design of a flexible and scalable interface with the access and transport domains of the network.

The first concept is based on the way that has been followed for the design of the service-based architecture (SBA) [9]. 3GPP defines the SBA as a set of functional components, known as interconnected network functions (NFs), where each one can use standardized interfaces, or service-based interfaces, to access and consume services of other NFs through an API-based internal communication. Towards 6G, the software industry investigates the capability to improve the modularity of services that are offered through the current SBA. In this context, a service can be broken down into fundamental service components, allowing third-party developers to mix and match components from different vendors into a single service chain.

For the second concept, programmability in the radio access domain is studied by the O-RAN Alliance, through the so-called functional split concept where the functionality of the RAN is softwarized and migrated to enable monitoring and automation services. In a similar way, for the transport domain, programming

7. Avnu Alliance® White Paper, "Industrial Wireless Time-Sensitive Networking: RFC on the Path Forward," Jan 2018.

8. IEC/IEEE 60802 TSN Profile for Industrial Automation, IEC CD/IEEE 802.1 TSN TG ballot.

languages like protocol-independent packet processors – P4 [2] can be exploited for end-to-end forwarding management and performance monitoring.

With the above-mentioned concepts, *the management capabilities at the service provisioning chain are expanded due to the usage of the cloud toolbox, while a use case-specific parading is born where third-party solutions can provide network automation and control solutions on top of flexible access and transport features (e.g., O-RAN and P4-programmability).*

7.2.2.1 Cloud-based services

The softwarization/cloudification of the NF (i.e., their implementation as virtual network functions (VNFs) or containerized network functions (CNFs)) is an evolution that brought cloud-based management services to the telecommunication sector. Indeed, extensive capabilities and tools primarily designed for cloud-native applications are provided based on the following aspects:

- Realization of Core NFs as Micro-Services: The realization of the 5GC NFs as micro-services (on containerization engines) provides capabilities, such as agile 5GC creation and flexible deployment, while it enables automated and lightweight lifecycle management. The main benefits are (i) the stateless implementation of the NF that facilitates the on-the-fly migration of the 5GC functions in different domains (e.g., to the edge) and (ii) the highly efficient sharing of the infrastructure resources. The cost for those benefits is the complexity of network-based chaining of the containers, as well as the additional computation cost required for the APIs to expose and consume processes.
- Adoption of Agile software development methods: This is a requirement of the new era in service provisioning, where the services should be able to continuously and easily update with improvements that reflect changing market demand. Such an approach is the DevOps methodology that integrates software development and IT operations.

In the same context, network slicing has emerged as a cutting-edge technology that allows for the creation of multiple virtual networks on top of shared physical infrastructure, allowing operators to provide portions of their networks that meet the needs of various vertical industries. A network slice is a collection of multiple sub-slices from various domains, such as the Core Datacentre, the transport network, and one or more edge locations. A key tool for the realization of network slices is the exploitation of ETSI VNF principles and existing frameworks, such as the Open-Source MANO[9] (an ETSI-hosted project to develop an Open Source

9. https://osm.etsi.org/

NFV Management and Orchestration software stack aligned with ETSI NFV). By providing specialized virtualized network slices for each vertical, network slicing will play a crucial role in addressing a variety of vertical applications. This is because new interactions and uses are anticipated to be brought about by the future ecosystem of smart connectivity. Section 7.3 provides a related solution, specified to the management of interconnected network slices though a logical network-as-a-service (LNaaS) approach.

Already network slicing tools have been released, such as the Katana Slice Manager[10] and the Open Slice framework.[11] In general, a network slice manager gets network slice template (NEST) for generating network slices through the north bound interface and offers the API for controlling and monitoring them. From Release 16 and onwards, 3GPP is working on providing higher flexibility and better modularization of the 5G System for the easier definition of different network slices and to enable better re-use of the defined services. Towards the enhanced service-based architecture (eSBA), as defined in Release 16, a service communication proxy (SCP) is introduced that can have a role in service selection, load balancing, and other common functions. This is an effort that supports the transformation of 5GC architecture to be as much compatible with cloud native as possible.

7.2.2.2 Procedure-based service structure – organic architecture

All the telecommunication network architectures were based on the concept of network functions as representing the basic functional element of a telecom network defined and standardized by its input interfaces, output interfaces, transfer function, and subscriber state. Ultimately, a network function is a system by itself that functions independently of other network functions which enables their parallel development and testing in isolation preparing them for system interoperability tests.

However, the complete independence of the network functions as part of the same subscriber connectivity service has significant limitations, which increase the complexity of the overall system, as illustrated in Figure 7.2:

- Maintaining a logically independent state in each of the network functions is creating an information multiplication as well as it requires many messages to be exchanged between the components, especially the acknowledgment that a specific step of a procedure was executed correctly. The additional steps create additional synchronization and scheduling functionality in the components themselves as well as prolong the end-to-end procedure duration.

10. https://github.com/medianetlab/katana-slice_manager

11. https://openslice.readthedocs.io/en/stable/

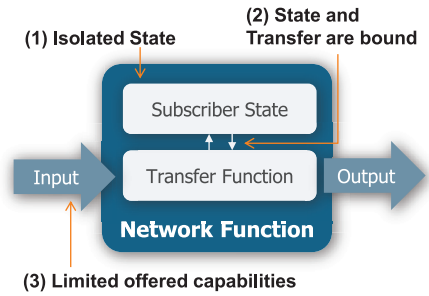

Figure 7.2. Network function concept elements.

- The state and the transfer functions are directly bound to each other. No matter if the subscriber state is grouped in a separate *Unstructured Data Storage Function* as in the 5G core network, it still must be fetched for at the beginning of the execution of each procedure step, modified, and pushed back at the end. These steps are executed in each of the network functions which receive the messages during the procedures inducing an additional execution delay.
- For the system to easily interoperate, the input interfaces must be clearly defined. This complete definition is also defining which type of messages can be received and processed. When aiming to introduce a new network function into the system, it must either use the existing input interfaces, thus the existing limited exposed functionality, or would require the modification of the other network functions. Considering that the 5G core network already has a very high split of functionality, when aiming to introduce a new service, represented by one or more new network functions, many interfaces would need to be modified. Due to this large entanglement, the system behaves highly monolithically in regard to adding new functionality.

However, as the core network functionality is expected to remain software only, the current network function concept does not have to be maintained. Instead, other concepts from Information Technologies (IT), specifically from web services can be adopted for a more flexible [10], reliable [11], and less complex network system [12]. Instead of splitting the functionality into network functions to describe different functionality in the core network, the network is split directly into services, as illustrated in Figure 7.3.

First, the core network is seen as a single large web service able to offer multiple services directly to subscribers. The user equipment (UE) is communicating with a single front-end component. The front end has very similar functionality to the proxy-call state control function (P-CSCF) in the IP multimedia system (IMS).

Instead of splitting the processing of the workers based on network functions, a new split is considered based on the procedure that has to be executed by the

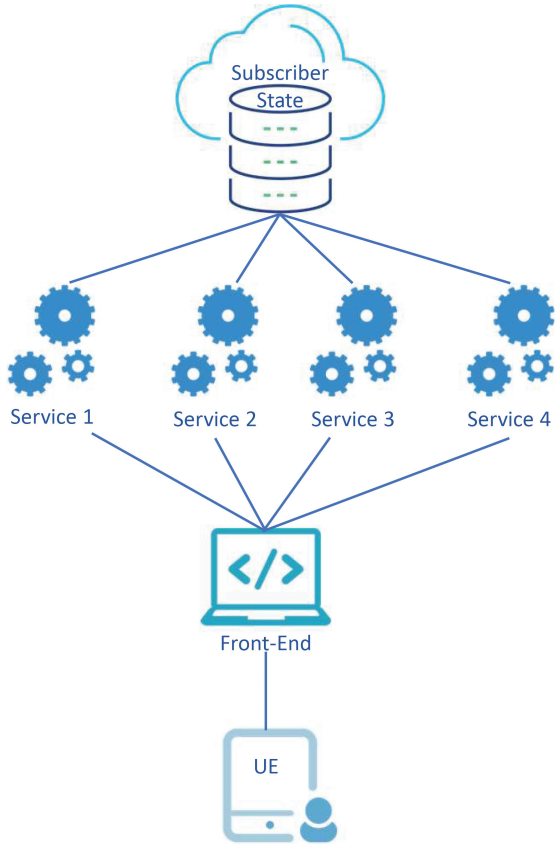

Figure 7.3. Services concept adapted for organic core networks.

system. Depending on the type of request transmitted by the subscriber, the front end will send the requests to a specific worker which will execute all the steps of a specific procedure. After receiving a request, the worker is fetching the subscriber state from the data base. Using the request and the subscriber state information, the worker will process all the steps of the request and generate new state information to be pushed to the database and the response to be transmitted to the subscriber.

Adapted to the current 5G network functionality, the concept will translate into the following functional split, illustrated in Figure 7.4.

Front end – a network component that receives all the requests from the UE. A security association is created with the FE during the initial authentication and authorization. This secure session is used to exchange all the messages of the UE. For this, a connection state is maintained with the UE. However, this state is independent of the connectivity service state as provided by the packet core. It represents only the secure association with the core network and not the subscriber connectivity session. And this could be skipped, although not recommended, if secure

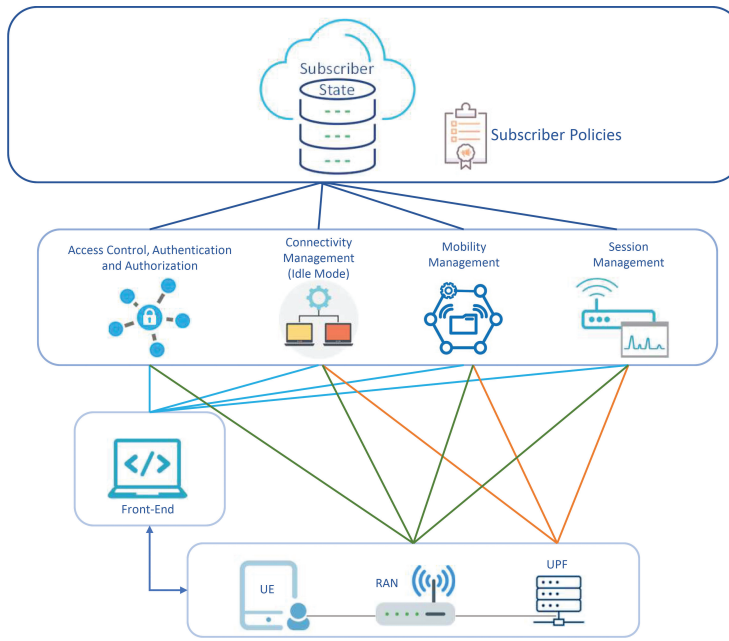

Figure 7.4. Organic core networks concept applied to 5G networks.

communication of the UE is not a requirement. The front-end role is like the access and mobility function (AMF) role in the 5G architecture in receiving the non-access stratum (NAS) messages of the UE. Also, it schedules the messages to specific workers, like the AMF for session management.

Workers – the workers are stateless components that execute the specific procedures of the core network and respond to the specific services. As a minimal example, the following basic services are considered:

- Access control, authentication, and authorization – this worker is executing the registration procedures with all their steps being equivalent to the 5G core AMF and authentication service function (AUSF) and to the 4G mobility management entity (MME).
- Connectivity management (idle mode) – a single service handling all the functionality related to the entering idle mode state, the subscriber-related tracking, service activations, and paging.
- Mobility management – facilitates the UE handover procedures similarly to the 5G NG Application Protocol (NGAP) UE. Compared to the 5G system, it is in charge of the execution of the handover procedures as well as the execution of the resource reservations during handovers, previously executed by the session management function (SMF).

- Session management – enables the resource reservations for the UE including dedicated resources. This worker is equivalent to the SMF main functionality.

Subscriber state – the subscriber state includes all the information that the core network maintains for a subscriber. With all the information maintained in the same place, there is no need for additional procedures of synchronizing state. This would significantly reduce the number of messages required for the communication.

7.2.2.3 Flexible and scalable management of access network

The potential to spit the radio access domain has emerged as an effort to softwarize as much as possible from the access functionality. This allows for (i) hosting centrally management functionality, a choice that can lead to an optimal (at the system level and not as conventionally at the base station level) management of multiple radio access nodes, (ii) supporting the networks slicing concept to the far edge of the service provisioning chain (access part of the network), and (iii) enable RAN management applications based on automatic control loops. Such an approach requires vendor-agnostic interoperability among different components, such as the central units, the distributed units, and the radio heads. In this direction, common interfaces are developed and open challenges are addressed by the O-RAN Alliance.

O-RAN Alliance was founded as a group of vendors, operators, and academic contributors towards producing standards to allow an inter-operable Open RAN deployment. It was formed by a merger between two different organizations, the C-RAN Alliance and XRAN forum. Three main control loops are defined in O-RAN architecture, as presented in [13]. They run in parallel and they interact with each other depending on the use case.

- Non-real-time RIC control loops have a timing of 1 second or more. rApps are used in this control loop as an independent software plug-in to provide extra functionality.
- Near-real-time RIC control loops have executing time between 10 ms and 1 second. xApps are used in this control loop to provide extra functionality.
- O-DU Control loops have a timing of less than 10 ms, but it should be noted that it is still not standardized as of yet by O-RAN, and the main focus is on the first two control loops.

Section 7.4 provides a representative solution for hardware and software programmability at the RAN domain, based on FlexRIC [14], which is in line in line with the reference O-RAN RIC approach, and provides better performance compared to the O-RAN's reference implementation.

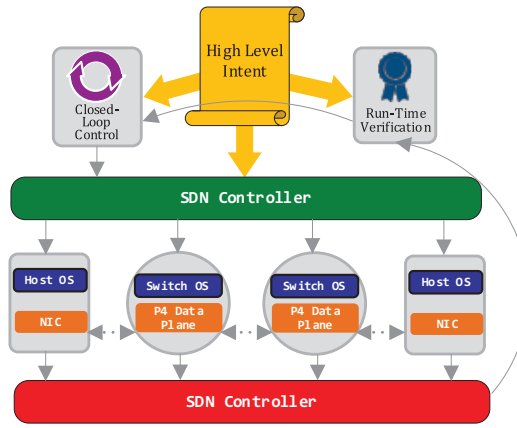

Figure 7.5. Reference architecture supporting closed-loop control with E2E programmability.

7.2.2.4 Flexible and scalable management of transport network

In an IP-based transport network, programmability could be applicable at the control and management planes, as well as at the data plane. For example, the function running on a programmable device can be changed from performing IPv4 routing to any other function simply by loading a new NF programme on the programmable networking device. This enables agile updating of functionalities executed on the UE and network side when programmability on these two ends is enabled.

Network deployments supporting closed-loop control with E2E programmability [15] can enable more effective resource management and operational robustness. P4-based E2E network programmability addresses packet forwarding policies as well as fine-grained telemetry. Fine-grained telemetry is supported by altering packets to contain information about the packet provenance (e.g., the path they took, the rules they followed, the delays they encountered, etc.). In network deployments supporting closed-loop control with E2E programmability, it is feasible to identify mistakes and make necessary repairs within milliseconds by keeping an eye on the network state and validating against packet telemetry.

A reference network architecture supporting closed-loop control with E2E programmability and fine-grained telemetry [15] is illustrated in Figure 7.5.

The elements of the architecture shown in Figure 7.5 and described in [15] are as follows:

- The data plane is made up of programmable switches and/or SmartNICss, which are usually realized using FPGA providing packet processing at line rate while being able to instantly update the forwarding behaviour in response to changes in the local state.

- The switch and host operating system includes the supporting IP network routing protocols (such as Open Shortest Path First (OSPF), P4 Runtime, and others), the associated algorithms, and the aggregation of measurement information received from the data plane.
- The SDN controller executes complex algorithms that would be challenging to represent as distributed computations while maintaining a network-wide perspective of the current topology and operational conditions.

The 5G System (5GS) of today already enables a number of closed-loop control use cases in the data plane. The above-mentioned fine-grained telemetry and UP programmability could further boost the effectiveness and performance of the network. These use cases are of interest:

- Traffic load-balancing control when multi-access protocol data unit (PDU) sessions are used supporting different access networks, including untrusted and trusted non-3GPP access networks, wireline 5G access networks, and so on.
- Traffic load-balancing control when redundant UP paths with dual connectivity (DC) are used.
- Traffic load-balancing control when the redundant transmission on N3/N9 interfaces is used.
- Control of ultra-reliable low-latency communication (URLLC) services.
- Decision for relocation of the UPF when acting as the PDU Session Anchor.

We believe that 6G will require full E2E programmability of the data plane with all the involved user plane nodes and the UE (controlled via NAS protocol). Currently, there are various technologies and programming abstractions that allow P4 E2E programming networks and end-hosts as explained in Section 7.5. Complementary to this, the open-source initiative by ETSI to further develop and evolve the TeraFlow SDN controller is presented in Section 7.3.

7.2.3 Enablers at the Service/Application Level

The openness at the service/application level is reflected in the wide adoption of APIs-based interaction in the telecommunication sector. Already, during the last decades, the use of APIs has served as a bridge between mobile operators and start-ups in emerging markets.[12] Operators have begun to consider whether to open their APIs, starting form APIs related to mobile messaging, operator billing, and so on. Irrefutably, this openness creates a powerful cycle of innovation as start-ups/third parties can combine several APIs to create new services. For example, a

12. https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2016/07/GSMA_Mobile-operators-start-ups-in-emerging-markets.pdf

start-up can offer SMS-based localized content to its users depending on their city or area and then charge them by deducting the amount from their mobile airtime. In the same direction, the TM Forum's 60+ REST-based Open APIs are developed collaboratively by TM Forum members working on the Open API project [16].

Overall, the availability/exposure of APIs from the network is the major enabler for network programmability, since it is a key necessity for any other technology enabler at management and connectivity layer described in previous sections.

At the network core domain, the network exposure function (NEF) of the SBA, exposes capabilities, provided by the other 5GC NFs, to external systems, i.e., NEF offers the appropriate APIs for the usage of SBIs from externals. More precisely, NEF enables external applications to communicate with the 5G core's SBA via APIs (i.e., Secure provision of information from an external application to the 3GPP network). Practically, it adapts and transforms telecom network interfaces to RESTful APIs. Considering the SBA, the main consumer of NEF APIs is the Application Function (AF). AF may or may not reside in the domain of the infrastructure owner (operator's domain), and its main functionality includes the provision of Packet-Flow Descriptors (PFDs) to NEF, and the consumption of RESTful APIs to utilize services and capabilities securely exposed by NEF. This openness of the 5GC provides third-party innovators (vertical industries) with the required toolset to (i) control the service deployment process, (ii) monitor the performance of their services, and (iii) feed 5GC NFs NF with information from the application layer.

The same scope of 5G-core openness to third parties is also served by the APIs that are developed on top of NEF, namely, the VAEs [17], the service enabler architecture layer (SEAL) services, as well as edge services through the multi-access edge computing (MEC) APIs. Verticals and operators can develop their own APIs and expose them securely by utilizing the 3GPP Common API Framework (CAPIF) [18].

Section 7.6 provides a representative development where an API programmability framework for interaction with the network core is provided as an open-source project.

7.3 Programmability Through ETSI TeraFlow SDN

This section includes anticipated service capabilities and concepts enabled by interconnected public and private networks in the context of 6G networks and slicing. The *Logical Networks as a Service* (*LNaaS*) concept and service model will become far more extensive, flexible, and pervasive as compared with 5G. While 5G is basically offering advanced connectivity service with the support of QoS from the UE/device to a data network (identified by a data network name), *with 6G, it is expected that a richer topology of connectivity can be supported, controlled and managed*

as a service, as an effort to support good or better customer/user experience, while achieving improved overall resource utilization and network performance. We consider even connections from/to multiple UEs/devices to/from multiple data network gateways and/or edge compute nodes/facilities. Complementary to the UE/device connection services, a mesh connectivity is expected among the edge compute facilities (cf. edge continuum) to support distributed application functions and services to the UEs/devices.

To support and complement this edge- and device-oriented connectivity, complementary connectivity capabilities are needed that will manage to reach remote end-points or vertical enterprise sites, in order to enable roaming services or third-party application services not located on the edge. For instance, there can be reached end-points located in the local operator network, or in a medium or long distance away across public interconnected networks. The above-mentioned enhanced connectivity can be based on basic Internet access services, or specialized connectivity services, whether enabled by a multi-mode extended Internet or some dedicated single or multi-stakeholder network complementing the current Internet.

We anticipate that the above will build from and extend the LNaaS concepts and offerings of 4G (based on the access point name, APN) and 5G (data network name, DNN, extending APN-based capabilities into a 5G slicing context) towards 6G, where the logical networks can be dedicated for an online application provider, a vertical enterprise customer (VEC) or established along with specific specialized application services offered by a CSP. Those logical networks should consider and enable a broad variety of advanced 5G and emerging 6G use cases, including integrated connectivity and compute service models and wireless end-points, where combined wireless communication and sensing is part of the 6G landscape, as presented in [19].

7.3.1 Transport Network Slice as a Service

Considering the LNaaS context, in the TeraFlow project,[13] logical networks are oriented towards VEC or OAP and they are supported and enabled by the concept of transport network slice as a service (TNSaaS). The TNSaaS concept must cater to elements and capabilities such as:

- Managed specialized connectivity services on-demand, with richer topologies and flexibility to accommodate the 6G-driven edge-cloud continuum.
- Supporting interconnection between private and public networks in various contexts and scenarios.

13. https://www.teraflow-h2020.eu/

- The UEs/devices can be attached to public networks, fully private networks, or private networks based on slicing in public networks; also, extending L2/L3 VPNs, in combination with the above.
- Including a variety of business model enablers to support existing and future business models.
- Supporting a variety of tenant networks, considering multi-stakeholder partnering, and security support.

All the above are enabled through further standardization and open-source initiatives, having as cornerstone SDN control and management, namely, the ETSI TeraFlow SDN (TFS) [20] controller, as described below. We consider the concept below as representing a baseline from which additional requirements driven by 6G can emerge while identifying gaps in today's standards and open-source initiatives.

ETSI TeraFlow SDN controller

TFS controller and the service concepts and information models developed in the Teraflow project are targeted to enable and facilitate logical network as a service to (vertical) enterprise customers as well as operator internal and inter-NSP connectivity and networking (cf. TNSaaS). TFS is constructed as an open-source software project within the ETSI community and is developed under an Apache2 license.

Figure 7.6 outlines the architecture and functional components of the TFS controller that will facilitate the deployment and operation of TNSaaS for Beyond 5G (B5G) 6G slices.

Several data models for services and devices were required to facilitate the request and tear-down of end-to-end slices for future applications and services. Also, industry standards and specifications are considered in the definition of the architecture of the TFS controller, crating in that way the potential for tangible impact in standardization and open-source communities. Figure 7.7 depicts related standard defining organizations (SDOs) and open-source software communities. It also highlights potential overlap among them.

Ongoing work is analysing and proposing service concepts and enablers to align ongoing work in different standardization camps, leveraging the core TeraFlow concept of transport network slice as a service, and developing the enabling functional components, APIs, and data models.

The TeraFlow SDO development activity is a parallel effort managed by technology experts from the project team. This activity has specific contributions in mind and participation in supporting TFS Controller development goals and interoperability plans; this SDO effort is categorized below with the technical area and objectives summarized:

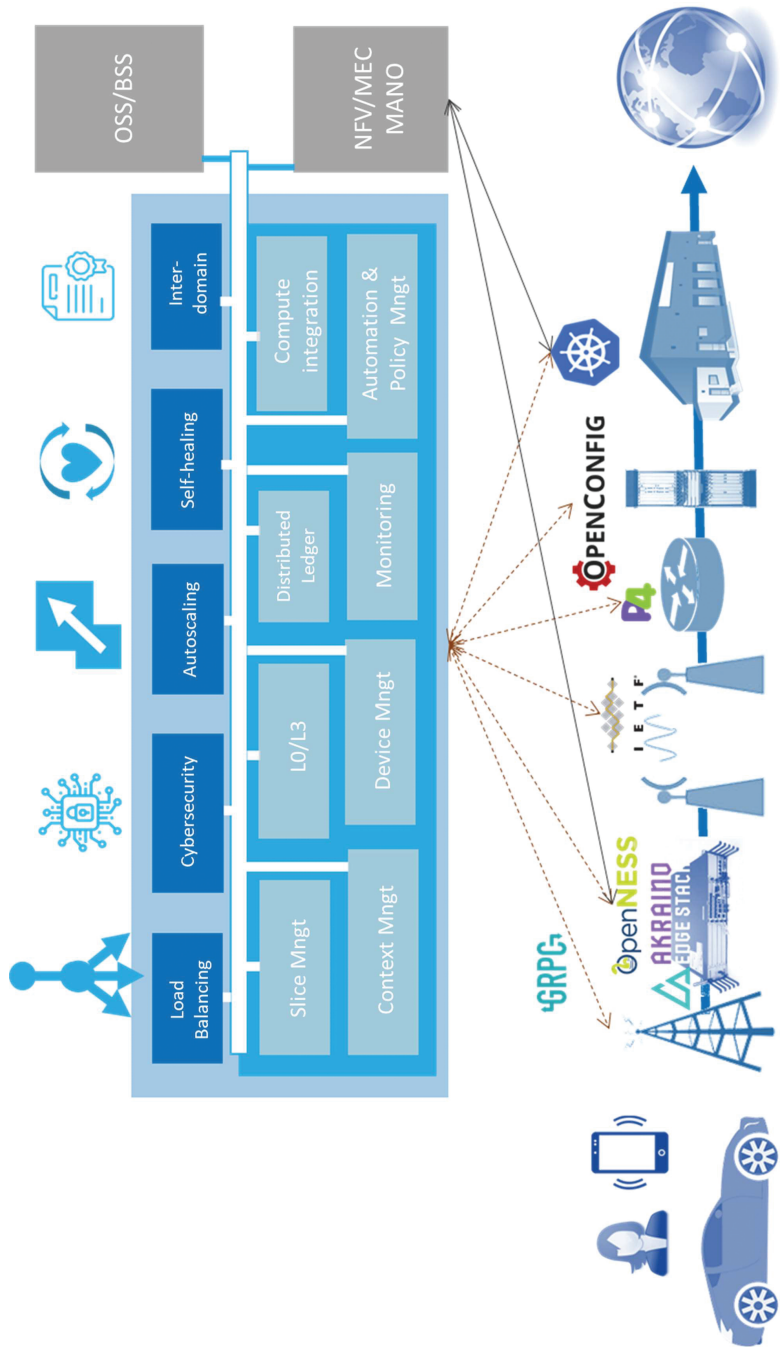

Figure 7.6 TFS controller architecture.

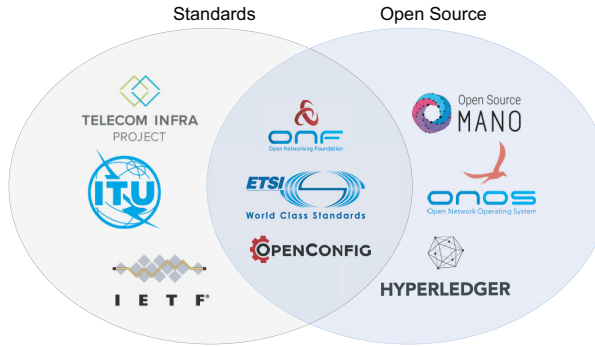

Figure 7.7. SDOs and open-source software communities related to the TFS controller.

- **TFS service and slicing models:** This IETF-based activity develops industry-recognized service models that will allow for end-to-end layer-3 and layer-2 network services to be requested. Additionally, the IETF has developed a network slicing framework. It establishes the general principles of network slicing in the IETF context and how it relates to non-IETF transport networks such as 3GPP network slices. TeraFlow project partners are authors of crucial service models [21], the project leads the editing and development of the IETF network slicing framework document [22].
- 3GPP Network Slices
  - Network operator internal slice to enable 5G operator services.
  - 5G slice is offered as a service.
- **Open networking foundation (ONF):** With a key objective of developing a cloud-native and scalable SDN controller (TFS), standardization efforts related to the development of data models and interfaces enable a hierarchy of controllers. In this sense, the ONF Open Transport Configuration and Control (OTCC) project aims to promote common configuration and control interfaces for transport networks in SDN. One of the project work items is the specification of the transport application programming interfaces (TAPI) data models, publishing open standard interfaces, whose main application domain is the controllers north bound interfaces. Lately, TeraFlow community has participated in the development of the TAPI Reference Implementation Agreement.
- **Telecom infra project:** The Open Optical and Packet Transport (OOPT) project of TIP works on the definition of open technologies, architectures, and interfaces in transport networks. Telefonica and SIAE are key members of TIP and active contributors in multiple OOPT subgroups.
- **Distributed ledger technologies:** The ETSI ISG PDL (Industry Specification Group Permissioned Distributed Ledger) facilitates the utilization of

blockchain technologies for the creation of open and trustworthy ecosystems of industrial digital solutions. TeraFlow project partners lead several activities in this ISG.

These open standards will need further development for B5G, 6G, and beyond.

In addition, as new use cases, requirements, and emerging applications and services are identified, new slice models and techniques must be considered. Initial considerations are provided in the fourth version of the 5GPPP White Paper on the 5G and beyond architecture.[14]

As 6G research continues, several emerging applications and services have been identified. These new types of network applications include:

- Tactile Internet of senses.
- Embedded intelligence and connected machines.
- Cognitive networks.
- Device and network twinning.

Investigation and discussion have already started, impacting how logical network slices are requested, designed, and deployed to support the previously mentioned 6G applications and services. There will be specific standards and open-source software required that would provide building blocks, including:

- Highly distributed Cloud-native infrastructure and orchestration, supporting massive scale rapid turn-up of container-based virtual functions.
- Location awareness and trust zones, as users and applications are attached to multiple network locations.
- Green and sustainable network technologies, especially the increased use of photonic systems (using up to 70% less power than electron-based communication).

Entirely new routing and addressing architectures may also be required to meet the demand of emerging 6G applications and services. For example, within the TeraFlow project, a study has been initiated on a new proposal called semantic routing and addressing [23], where packet forwarding and path selection decisions are based on contextual information carried in the packet header.

7.4 Programmability Through O-RAN-Compliant SDK

Software- and/or hardware-based solutions operating in the RAN, edge, transport, and core segments of E2E 5G/6G infrastructure provide programmable data path

14. https://5g-ppp.eu/wp-content/uploads/2021/11/Architecture-WP-V4.0-final.pdf

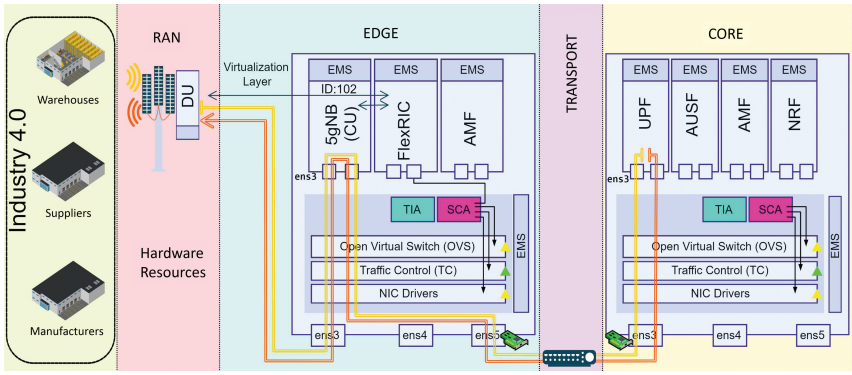

Figure 7.8. E2E programmable data path and data path network slicing.

and advanced network slicing capabilities, which allow user-definable flexible classification and prioritization of the traffic in complex networking environments featuring 5G/6G encapsulation overlays and industrial protocols such as EtherCAT and so on due to vertical business use cases and guarantee the QoS of the prioritized traffic accordingly. In this context, software solutions based on Flexible RAN Intelligent Controller (FlexRIC), Traffic Control (TC), Open vSwitch (OVS), and so on can offer cost-efficient programmable data path capabilities in the kernel of the system, while Smart Network Interface Cards (SmartNICs) such as NetFPGA and Netronome can provide performance boost benefiting from hardware acceleration. Furthermore, software- and hardware-based solutions can be exploited in a hybrid manner wherever appropriate along the E2E path. Figure 7.8 shows an architectural view of an E2E programmable data path and corresponding data path network slicing in 5G and beyond networks for Industry 4.0 and other use cases.

In this architecture, FlexRIC, TC, OVS, and SmartNICs operating in the data plane achieve E2E network slicing by leveraging the programmability of these data path components.

FlexRIC [24] represents a software-defined RAN controller, built through a server library, controller-internal Applications, and optionally a communication interface, all offered by the FlexRIC SDK. The FlexRIC SDK consists of server and agent libraries that are used to build *controller-internal applications*, which, on the one hand, help specialize the RAN controller towards specific use cases and, on the other hand, enable *external applications* (xApps) to conveniently control different RAN functions. In the case of network slicing, the fast control loop of FlexRIC can enable reinforcement learning-based xApps to correctly follow their Markov decision process modelling of the RAN slice scheduling problem, as if they are running in a "zero delay state-to-action" simulation. Furthermore, FlexRIC convenient scalability in supporting multiple xApps of different languages also allows developers to conduct complex orchestration of multiple components, such as machine learning

operations. Together, developers can expect to have their simulation-trained xApps ready for production. Overall, *the controller-internal applications allow modularly building specialized RAN controllers for specific use cases, and optionally, they can expose information to external applications (xApps) via a northbound communication interface to allow the xApps to control the RAN in line with the reference O-RAN RIC approach*.

TC, OVS, and SmartNICs represent controllers for the non-RAN segments (edge, transport, and core networks). TC [25] is a Linux user-space utility programme for configuring the Linux kernel packet scheduler. The scheduler with advanced queuing disciplines and dedicated filtering expressions (covering 6G network traffic) can optimize and guarantee performance, reduce latency, and increase usable bandwidth for services with specific network requirements. TC is recommended in systems that are not equipped with higher-performance programmable data-plane network devices such as those based on SmartNICs or kernel bypass techniques, or in those scenarios where the link (L2) and network (L3) layers of a resource are required to work together. OVS [26] is a software switch in virtualized network environments, and it can be extended to allow 5G/6G-compatible data path network slicing with customized network slicing extension profiles and network slicing extension features. Experimental results show that this approach is able to provide connectivity for up to 1 million IoT devices in mMTC traffic, achieve over 19 Gbps bandwidth in congested eMBB scenarios, or ensure delays in the order of μs for critical-missions communications, providing high reliability in all tested scenarios (0% packet loss ratio). As to SmartNIC-based controllers, NetFPGA [27] and Netronome [28] are promising platforms, among others. For instance, NetFPGA can be explored to achieve a data path network slicing enabler based on the SimpleSumeSwitch model and by leveraging the P4-NetFPGA project. Performance enhancements can be expected in certain use cases thanks to hardware-based acceleration, as reported in [29].

In the control plane, the *Slice Controller Agent* coordinates the various data plane controllers in collaboration with the network slice management plane (not shown in the figure), and it is network topology aware with the help of the *Topology Inventory Agent* to initiate and control the network segment specific controllers. More details of controller-specific implementation architectures can be found in [30].

7.5 P4-Based Framework for E2E Programmability

7.5.1 Network Programmability with P4

We choose the P4 language [2] as an example of programming abstraction to show the benefits that can be achieved when programmability is employed in networks.

The P4 language is one initiative to abstract the packet processing pipeline of network devices and flexibly define its behaviour. P4 is platform-independent, meaning that it can be used to programme various classes of packet processors such as software switches, SmartNICs (NIC stands for network interface card), field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs). These different classes of packet processors can be used to build the infrastructure of the next cellular generation clouds.

7.5.1.1 Performance-aware management of programmable networks

When using programmable networking devices, it is important to understand their forwarding performance and to guarantee certain performance levels. This is achievable when utilizing models that can predict the packet forwarding latency when running any P4 programme on arbitrary P4 devices as it has been proposed in [31]. This model is based on an extensive measurement campaign that identifies the base processing delay of different P4 devices, in addition to the marginal delay of executing atomic P4 operations on these devices. First, performance profiles for different types of packet processors are created using the collected measurement results. Then, any given P4 programme is decomposed to its atomic operations whose processing latency is already quantified and recorded in the pre-developed performance profiles. For example, in the case of IPv4 Forwarding NF, these atomic operations are extracting and manipulating Ethernet and IPv4 headers. Finally, the performance model estimates the packet forwarding latency when running arbitrary NFs on any P4 device as the sum of the base processing delay of that device and the marginal latency cost of executing the NF's constituent atomic P4 constructs. The model showed sub-microsecond prediction accuracy when tested for different NFs and P4 devices. To optimize the management of programmable networks, the optimal placement of NF workloads on heterogeneous programmable substrates should be investigated. We propose in [32] a mathematical formulation for an optimization problem that aims to optimally place different NFs into P4-based cloud environments. The network orchestrator can use the performance model described above in [31] to perform the placement in a performance-aware manner. This allows for achieving the highest levels of performance when managing these programmable networks and meeting QoS requirements. The problem formulation takes into account various capabilities and characteristics of the hosting P4 device. These device characteristics include the supported P4 architecture, supported extern functions (i.e., non-native P4 functions such as cryptography), basic processing latency, marginal latency for executing atomic P4 constructs, latency for executing P4 extern functions, throughput capacity, processing resources availability, and cost.

In addition, the problem formulation considers the NF workload's requirements in terms of desired throughput, desired P4 architecture, desired P4 extern functions, and constituent P4 constructs.

For a given NF workload, the objective function is to find the optimal set of P4 devices and the optimal embedding of NFs into these devices while minimizing both the total forwarding latency in the system and the capital cost for building the system. The performance model described before (as that in [31]) is used to calculate *a priori* the delay that results from different placement options as shown in the following equation:

$$\Delta_d^f = \delta_d^{BP} + \sum_{c \in C_f} \delta_d^c + \sum_{e \in E_f} \delta_d^e$$

where Δ_d^f is the forwarding latency when running NF f on P4 device d. It is equal to the sum of three components: (i) the base processing latency on P4 device d, denoted as δ_d^{BP}; (ii) the sum of the marginal latency when running the atomic P4 constructs $c \in C_f$ that constitute NF f on P4 device d denoted by δ_d^c; and (iii) the sum of the latency of extern functions $e \in E_f$ required by NF f denoted by δ_d^e.

Several constraints should be satisfied. These include the following: (i) an NF should be placed on a P4 device only if the device supports a compatible P4 architecture and includes all the extern functions required by the NF; (ii) the sum of throughput required by different NFs to be placed on a device shall not surpass the limited throughput of that device; (iii) the processing resources capacity of a P4 device shall not be surpassed; and (iv) some NFs like the IPv4 forwarding NF must be placed on every used device to guarantee proper packet forwarding between devices, and so on.

To evaluate the proposed workflow, the target use case includes a combination of different types of P4 devices to build a programmable substrate for a cloud environment. These devices belong to different classes of processing platforms such as CPU, FPGA, NPU (network processing unit), and ASIC. The capabilities and performance of these devices are already studied in the literature, and a summary of these findings can be found in [32]. Figure 7.9 depicts a web chart that summarizes the comparative advantages of different P4 device types based on different criteria. A set of common NFs such as IPv4 forwarding load balancer, and so on are selected as the workload to be handled by the network. More details about the evaluation in terms of the requirements of the used NFs and the capabilities of the selected P4 devices can be found in [32].

Four scenarios are defined and evaluated wherein the weights of the two selected objective functions vary (i.e., the forwarding delay in the system and the cost of building the system). Scenario 1 (S1) targets achieving the best

performance without worrying about costs, while Scenario 2 (S2) targets finding the cheapest solution where best-effort performance is sufficient. Scenario 3 (S3) targets achieving a balanced solution in terms of the best performance and minimum costs. Finally, Scenario 4 (S4) targets achieving the best performance under the limited budget of $100k.

The placement results for Scenarios 1, 2, and 3 are trivial where the optimal solution included an increasing number of a homogeneous set of devices as the workload increases. The ASIC devices with the highest performance were selected in S1, while the CPU-based devices were selected in S2 as they are the cheapest solution. In S3, the NPU-based devices were selected since they achieve the best trade-off between cost and performance.

The placement results of S4, wherein a cost limit of $100k is defined, are more interesting to analyse. Figure 7.10 depicts the results corresponding to this scenario showing the number of instances of each device type required for an increasing number of NFs to be placed. When the workload is low, a single ASIC device is sufficient to handle the load while providing the best performance. When up to 22 NFs must be placed, another ASIC device is needed to handle the load since the first device's processing resources are exhausted. As the load increases further, no more ASIC devices could be utilized since the remaining budget only allows for the second-best performing device, which is in this case FPGA-based. At this stage, up to four FPGA devices are required to process the increased load until reaching a total of 90 NFs. Afterward, one of the two ASIC devices is discarded to afford to employ a bigger number of cheaper FPGA devices to handle the increased workload. When the number of NFs to be deployed reaches 174, the second ASIC device is also sacrificed and replaced with additional FPGAs whose number increases to 20 FPGAs when the number of NFs workload reaches 200. After this point, the optimal solution tends to further sacrifice performance through replacing FPGA devices with the next best performant device, i.e., NPUs, to enable handling the increased number of NFs within the available budget.

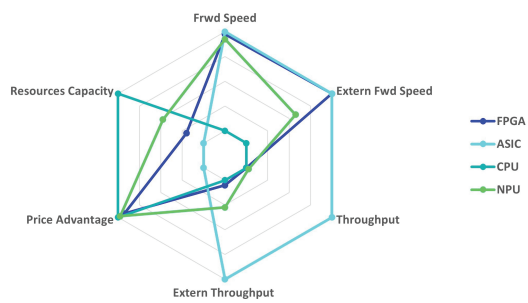

Figure 7.9. Comparative advantage of P4 device types.

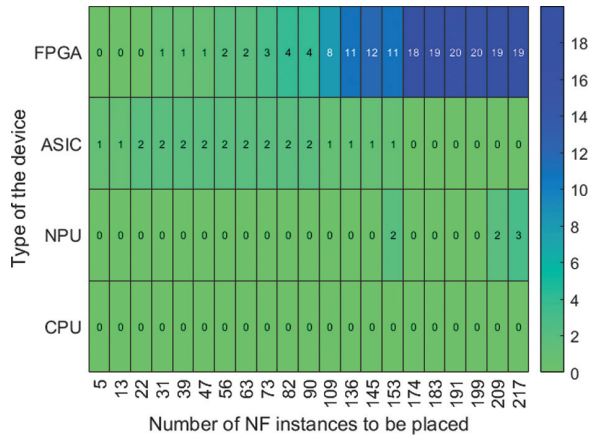

Figure 7.10. Used P4 devices in Scenario 4.

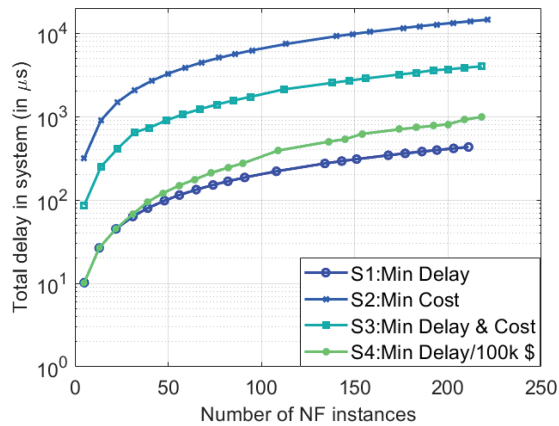

Figure 7.11. Total forwarding delay in the system.

The delay and cost functions of the optimal solution are depicted in Figures 7.11 and 7.12, respectively, for various scenarios as a function of the number of NFs to be placed. Following the defined objective, the overall delay in S1 is minimal, while the overall cost in S2 is the lowest. The results for S3 reveal the trade-off between the two objective functions, where the overall delay and cost of the system are both minimized. The results of S4 show that the delay in the system is as low as that of S1 (when only the delay is minimized) until the limit on the budget is reached after 22 NFs. After this point, the system's delay begins to increase diverging from the delay in S1, while the cost stays always below the limited budget of $100k.

In summary, utilizing precise performance models for programmable network devices and intelligently managing these devices enables the creation of flexible networks without sacrificing performance. Additionally, proper consideration of

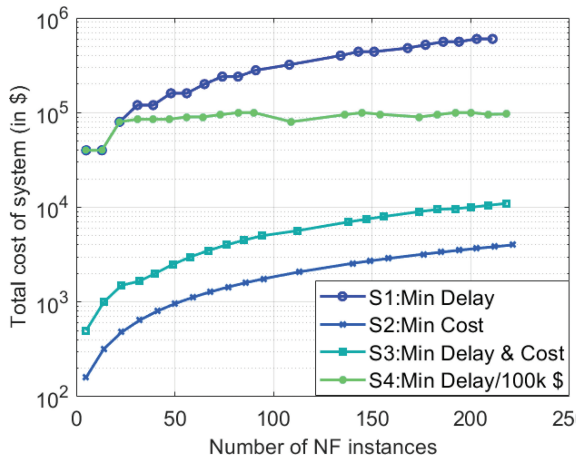

Figure 7.12. Total cost of the system.

the network substrate's capabilities and costs enables cost-effective infrastructure planning to reduce the system's total cost of ownership.

7.5.2 Extensions Towards UE Programmability

The standardization process in 3GPP time and time has proven its paramount value for numerous successful generations of cellular networks. There are, however, some hiccups: the process is time-consuming; some features are niche and do not interest all members; it takes years for a feature to be introduced and there is no guarantee it will be implemented. To some extent, this could become a barrier to innovation, especially when considering aspects, such as network flexibility for different deployments and use cases. For 6G, the UE programmability concept aims to significantly decrease the time to innovation for features having an impact on the air interface protocols. The concept refers to defining API(s) for the UE associated with actions/routines/sub-routines that can be exposed to a programmer entity so that a programmer entity is able to modify/add one or multiple behaviour(s) at the UE associated to the air interface protocols. A high-level signalling diagram and architecture are shown in Figure 7.13.

As a result, with a reduced amount of 3GPP standardization, the programmer entity should be able to define new behaviours/features for the programmed UE, such as a new message received or transmitted, new information elements and associated interpretation, new reports, new trigger for that message or report, and new/additional triggers for existing messages.

At a high level, some initial components are essential to enable UE programmability. Those components can be considered high-level solutions that are needed initially to realize the concept. Here, three of those are provided, which are API

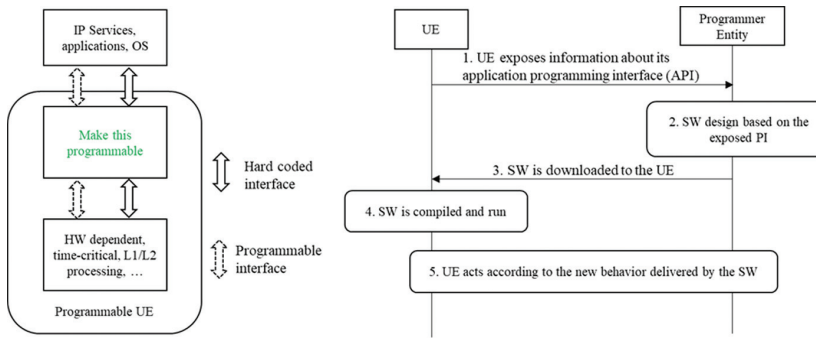

Figure 7.13. To the left, high-level architecture of a programmable UE, to the right, signalling diagram for programmability.

exposure towards the network, initial access mechanism for programmable UEs, and software version management.

The API exposure mechanism is needed so that the programmer entity knows the specific capabilities of programmable UEs in order to utilize their capability. Hence, there needs to be a signalling procedure to convey the programmable capabilities and their specifics to the programmer entity. Such capability exposure can be initiated at specific events such as UE registering with the network or can be on-demand using dedicated signalling towards specific UEs. The capabilities regarding programmability can be standardized to include specific APIs and UEs can indicate which subset of standardized APIs are supported. Therefore, the programmer entity receiving the capability can design suitable SW for specific UEs based on available APIs.

Upon initial access, programmable UEs may have an SW version for a specific behaviour that is different from the one on the network side. Hence, the UE may not be able to connect to the network because of SW incompatibility. A bootstrapping broadcast channel can potentially solve this problem by providing information on the current version of the SW on the network side and how to acquire it. A programmable UE upon initial access first checks the bootstrap channel to acquire information on the active SWs and how to acquire them. Based on such information, the UE can download the proper SW version and hence connect to the network.

Finally, SW management mechanisms are needed because for specific behaviour implemented by an SW, there could be various versions corresponding to new updates or different interests of operators/vendors. Hence, an SW management solution is required to enable synchronized operation of the UE and network with respect to SW versions. Such a solution will serve to store and manage multiple SW versions including adding, removing, and updating SW. Moreover, the solution will allow to select a specific SW version to be initialized upon request from the network.

However, there are many challenges to overcome for the successful adoption of the concept which faces manifold open research questions.

One aspect is to ensure that the programmability solution does not disrupt 3GPP. As mentioned before, the 3GPP way of working is vital to the success of cellular network evolution and is trusted by every player in the ecosystem. The concept must be developed in harmony with the 3GPP and complement it rather than disrupting an already successful framework. This can be achieved by defining an overall framework for UE programmability by defining a bare minimum for the concept in 3GPP, such as methods of downloading a software and defining and exposing APIs.

Another challenge is that there are potentially many flavours of programmability, each with advantages and disadvantages. Each flavour balances a trade-off between its capability and pragmatism. At one extreme edge, one can envision a downloadable UE stack paradigm potentially offering full programmability. However, developing this approach from concept to reality will face many difficulties ranging from technical issues, split of responsibilities and concerns among different entities/vendors, trust and privacy issues, and acceptance from 3GPP. On the other hand, the programmability could be introduced in a limited way by allowing only specific features, e.g., radio resource management measurement, to be programmed. Such an approach will be more acceptable from the point of pragmatism at the risk of being very limited and potentially leading to the introduction of multiple APIs per protocol stack.

Another challenge is related to a typical UE hardware. The UE hardware typically has a small footprint and is packed with optimized codes to achieve extreme efficiency in contrast to more flexible hardware such as a VM. Thus, any framework should make an extra effort to accommodate this constraint.

Privacy and security aspects should be considered fundamental parts of the concept. A security/privacy functionality needs to ensure that UE always receives a safe programme, the received programme does not introduce security concerns, the privacy of the UE is never compromised, and UE is implemented with a mechanism to ensure trusted computing.

Another challenge is considering the mobility aspects. The programmability framework should not restrict the UE from moving freely in the network. When a new behaviour is programmed to the UE, e.g., by an SW patch, then the framework should ensure that the UE is not interrupted when moving to another part of the network because either that specific SW is not available or have non-compatible versions. This also raises the multivendor issues and the need to properly handle trustworthiness aspects when it comes to code exposure.

Beyond the high-level solutions presented here, enabling the realization of the concept in a general framework, the next step is to develop a concrete architecture

for the UE programmability and specific use cases that it can realize for a programmable configuration of the air interface. The introduction of conditional handover [33] in NR enables the UE to participate in the decision to reconfigure the air interface and this facilitates to pursue a programmable reconfiguration use case.

7.6 Programmability Through the 3GPP API Framework

API-based interaction of third parties with the network is needed for the support of openness at deployment, management, and application levels. In this context, the development of a Common API framework (CAPIF) has been coined in 3GPP as an effort to avoid duplication and inconsistency between the various existing API specifications. As such, 3GPP CAPIF includes, under a common architecture, aspects that are applicable to any service APIs at the northbound of a mobile network. More precisely, CAPIF is a complete 3GPP API framework that covers functionality related to on-board and off-board API invokers, register and release APIs that need to be exposed, discovering APIs by third entities, as well as authorization and authentication. From the market perspective, the need for such a management framework has well recognized, while the CAPIF implementations[15] and products are still under development. For instance, proprietary solutions have emerged, such as the Red Hat API management.[16] It is based on the 3scale enterprise API management product of the company, and it provides authentication, governance, security, analytics, automated documentation, developer portals, and monetization for API services.

The 3GPP CAPIF functional architecture is covered in 3GPP TS 23.222 [18, 34] "Functional architecture and information flows to support Common API Framework for 3GPP Northbound APIs" (since Release 15). Based on the architecture and the procedures defined in these reports, additional information and requirements are considered, regarding CAPIF security features and security mechanisms, as presented in 3GPP TS 33.122 [35]. The specification of the CAPIF APIs that are needed for realizing the CAPIF functionality is part of 3GPP TS 29.222 [36]. Actually, within this technical specification, the interacting protocol for the CAPIF Northbound APIs is described.

CAPIF functionality is considered a cornerstone in the realization of mobile network openness, since it allows secure exposure of core network APIs to third-party domains, and also, enables third parties to define and expose their own APIs.

15. https://github.com/EVOLVED-5G/CAPIF_API_Services

16. https://www.redhat.com/en/blog/api-management-3scale-service-provider-use-case

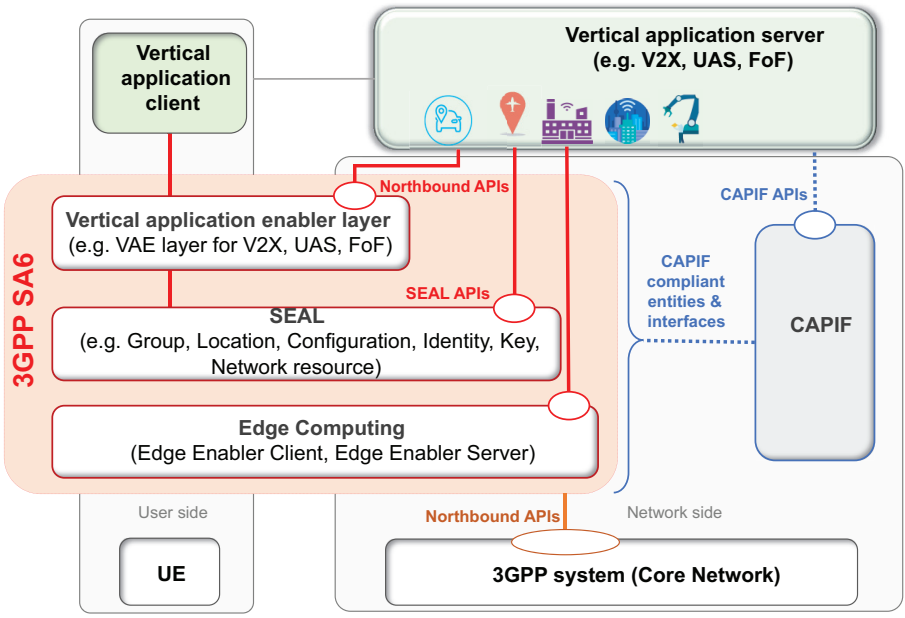

Figure 7.14. CAPIF in the context of 3GPP SA6 activities.

Indeed, CAPIF has become already a fundamental feature for the 3GPP SA6, targeting the interaction of Verticals with the 5G system. In this context, 3GPP introduced the concept of VAEs, enabling the efficient use and deployment of vertical apps over 3GPP systems. The specifications and the architecture are based on the notion of the VAE layer that interfaces with one or more Vertical apps. VAEs communicate via network-based interfaces that are well-defined and version-controlled. The focus of VAEs is to provide key capabilities, such as message distribution, service continuity, application resource management, dynamic group management, and vertical app server APIs over the 3GPP system capabilities. The importance of realizing CAPIF is reflected in the fact that CAPIF compliance is required (Figure 7.14) in (i) the development of VAEs for various vertical industries (V2X, Factories of the Future, etc.), (ii) the realization of the service enabled architecture layer (SEAL), as well as (iii) the implementation of the service side of edge computing services.

7.6.1 CAPIF Services and Implementation

CAPIF architecture is presented in 3GPP TS 23.222 [18] and includes three main entities, namely, the API invoker, the CAPIF core function, and the API provider.

The API invoker is typically provided by a third-party application that supports capabilities, such as supporting the authentication by providing the API invoker

identity and other information required for authentication of the API invoker and discovering service API information.

The CAPIF core function is the main entity of the CAPIF and it consists of engines that among other capabilities authenticate the API invoker based on identity and/or other information, authorize API invokers prior to accessing service APIs, on-board/off-board API invokers, monitor service API invocations, and store policy configurations related to CAPIF and service APIs.

The API provider is an entity that provides API exposing, publishing, and management functions.

- The *API exposing function (AEF)* is the provider of the service APIs and is also the service communication entry point of the service API to the API invokers.
- The *API publishing function (APF)* enables the API provider to publish the service API information in order to enable the discovery of service APIs by the API invoker.
- The *API management function (AMF)* enables the API provider to perform administration of the service APIs.

Based on the three fundamental entities that CAPIF architecture has defined, a set of reference points (interfaces), with associated management APIs, for enabling the interaction between API Invokers and AEFs, are specified as well.

To facilitate any further contribution in the area, the currently available open-source implementation of CAPIF[17] (as it is being developed in the EVOLVED-5G project by Telefonica and Fogus innovation and services[18]) [36] follows the principles of microservice programming, and it is released together with a set of evaluation tests. The major aspects of this implementation are further described below.

Based on the CAPIF architecture, the core part of CAPIF is the CAPIF Core Function (CCF). To implement the API services of the CCF, the first thing needed is the CAPIF API definitions/signatures. Those have been specified in 3GPP TS 29.222 [37], and 3GPP has published the related YAML files[19] as well. The following services have been defined for the CCF:

- Discover Service: API to ask CCF the list of APIs published and available in CAPIF.
- Publish Service: API to publish API information from APF/AEFs.

17. https://github.com/EVOLVED-5G/CAPIF_API_Services

18. https://fogus.gr/

19. https://forge.3gpp.org/rep/all/5G_APIs
https://github.com/jdegre/5GC_APIs

- Events: API to manage Event subscriptions that enable event notification from CCF.
- API Invoker Management: API to enable the onboarding of API Invokers into CCF.
- Security: API to enable setting security profiles and retrieve security Tokens.
- Access Control Policy: API to manage access control rules in CCF.
- Logging API Invocation: API to add logs on API consumption.
- Auditing: API to query and retrieve service API invocation logs stored on the CAPIF core function.
- AEF Authentication: API for AEF security management.
- API Provider Management: API for API provider domain functions management.
- Routing Information: API to provide API routing information.

The YAML files of those services can be used by a Swagger editor to inspect all information elements in JSON. Each of these YAML files defines one or more APIs and the supported methods to use them (POST, GET, DELETE, and PUT). With the YAML files of the services described above, it is possible to generate automatic code that implements HTTP/HTTPS Endpoints that act as Servers that accept HTTP requests.

To build the CAPIF core function, several software tools have been used to develop, implement, build, and test the CAPIF API services.

- **OpenAPI Generator[20]:** This software programme allows the generation of API clients SDKs (Software Development Kit tools) or API servers given an OpenAPI specification. Moreover, it is possible to generate code in more than 20 different programming languages.
- **MongoDB:** Mongo is a non-SQL and open-source database tool used to provide storage to different CAPIF core functionalities.
- **Nginx:** Nginx is an open-source web serving technology that is used as a reverse proxy to forward requests to the different CAPIF modules.
- **Flask:** Flask is a micro-service web framework written in python used to build CAPIF. The main advantage of this framework is its modularity. This feature allows us to run different CAPIF services and mix them directly with other services as database with relative ease.
- **Robot framework:** Robot is a generic test automation framework used for acceptance testing and acceptance test-driven development.

20. https://github.com/OpenAPITools/openapi-generator

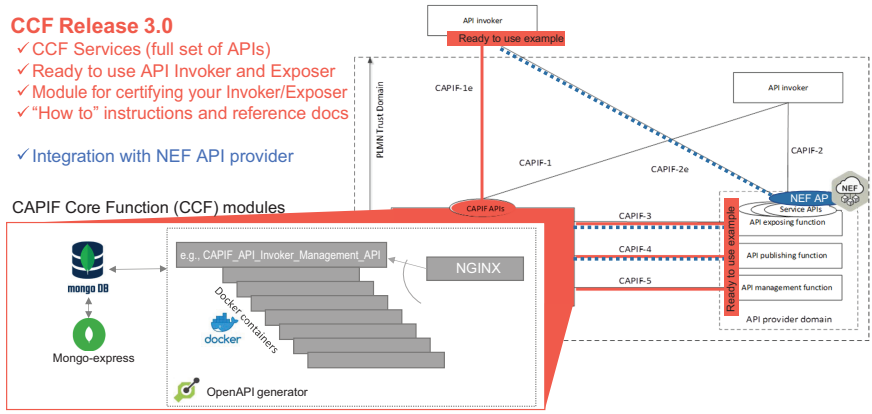

Figure 7.15. Open-source implementation of the 3GPP CCF (Release 3.0 features).

- **Docker:** Docker is an open-source containerization tool for building, running, and managing containers, where software is deployed. We use Docker to build each service of the CAPIF. In this way, each CAPIF service development is kept isolated from other services. This design pattern is known as a microservices-oriented architecture and is widely used in software development due to its innumerable advantages like better fault isolation and improved scalability, among others.

In order to guarantee the integrity of our implementation, an automated test suite is used, covering the core functionality of each CAPIF API service. Robot framework is the testing tool that ensures the quality and robustness of developed code. Moreover, it is defined as a test strategy to improve the code quality. This test strategy is composed of two steps:

- **Test plan document elaboration:** In this step, test plans are described, including various behaviour scenarios (considering both success and failure cases). The test plan structure includes clarifications on the pre-conditions, the action that takes place, and the post-conditions (response/result expected). The horizon of the test plans that can be defined, moves beyond the request–respond information that is available in the related 3GPP specifications, in a sense that behavioural scenarios beyond the basic functionality are defined to stress test the implementation integrity.
- **Test implementation and execution:** This step continues after finishing the elaboration of the test plan documentation. Each test suite is implemented and included in an automation pipeline that checks the status of the code after every deployment in the platform.

7.6.2 NEF as API Exposing Function

One first example of the exposing function is hiding all the underlying topology of the core network and playing the role of API provider, as defined in the CAPIF architecture. In future systems, this role can be played by other functions that refer to core, transport, or radio domain, and expose APIs for interaction from control, data, or management plane. With no loss of generality, here further analysis is provided for the case of network exposing function (NEF). NEF comprises several services that can be described as monitoring services, policy and charging services, application provisioning services, analytics services, Industry 4.0/IoT (Internet of Things) specific services, and security services.

The exploitation of the NEF capabilities from industry has begun already; however, before taking full advantage of NEF-based network exposure, many challenges are to be addressed. Indeed, topics regarding the exposure capabilities of the network are still to be considered in 3GPP Release 18, while telecommunication vendors are working to adapt the SBA and implement the already specified exposure functionalities.

To enable interaction with the core network, the implementation of NEF functionality as an open tool is a prerequisite since currently there are no open commercial solutions implementing the entire SBA and the southbound interfaces that NEF requires in order to expose the standardized APIs. Nevertheless, a NEF Simulator [38] has been developed by NCSR "Demokritos" aiming at surpassing this challenge by creating simulated and emulated events. The architecture of the Simulator[21] is decomposed into three distinguishing parts depicted in Figure 7.16.

The main features of the NEF architecture are described below:

- **Exposure layer (NEF APIs):** The principal idea of the simulator is the provision of the APIs that 5GC's exposure function (i.e., NEF) defines. Therefore, the available APIs have been placed in the 5G Exposure layer. Currently, the available APIs include monitoring events (i.e., location) and session establishment with QoS. As the work progresses, new APIs will be gradually added to the simulator and the existing ones will be enhanced.
- **Simulation environment:** As mentioned above, currently, communication with the southbound interface (i.e., 5GC) is a demanding task. However, the simulator can tackle this challenge by creating simulated events. To achieve this, it provides an interactive geolocated environment where users can create different network scenarios. These scenarios are designed to simulate the basic aspects of a 5G network, required for testing the available service APIs. For

21. https://github.com/medianetlab/NEF_emulator

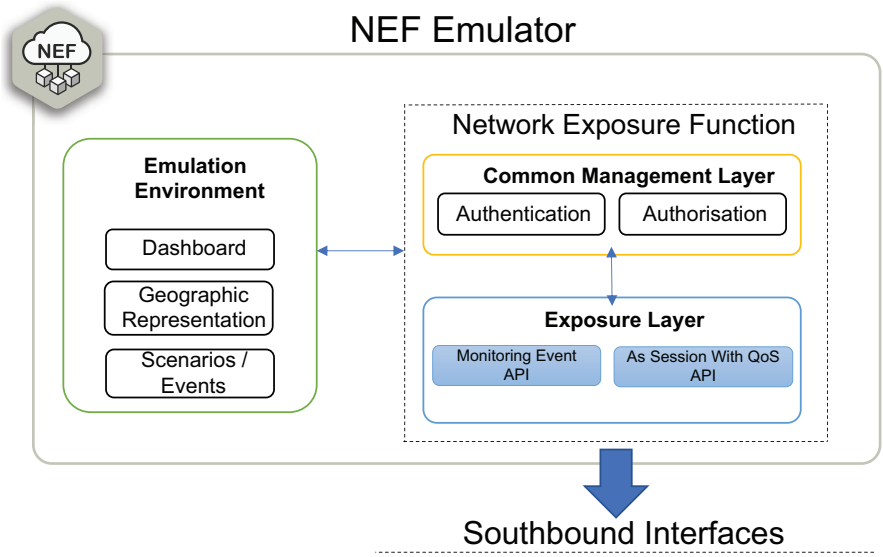

Figure 7.16. NEF simulator architecture.

example, in order to retrieve the location (i.e., cell level accuracy) of the UEs through the Monitoring Event API that NEF exposes, the simulator allows for the implementation of a scenario where UEs are moving through 5G cells. Developers are able to alter data, allowing them to define and run specific scenarios according to their needs.

- **Common management layer:** The simulator also provides common management functions such as token-based user authentication/authorization. Initially, to gain access to the simulator, there is a need for the user to create an account. After the creation of the account, an authorization step based on OAuth2.0 takes place, in order to make use of the available Northbound APIs or the Simulation Environment. Each application developer has a registered account/profile within the simulator that is considered an isolated environment; thus, NEF Simulator can store different scenarios configured by multiple users.

7.7 Programmability Enables the Network App Ecosystem

From the business perspective, programmability enables a new business potential around the development of the so-called network applications. The major challenge for the Network Apps ecosystem lies in the need for continuous development, test, and evaluation of vertical-specific network-enabled applications, on top of realistic

configurable infrastructures, prior to their commercial deployment in the mobile networks (operators' infrastructures).

As a response to this challenge, a facility that could support in long term the vertical application development and provisioning over mobile networks is required. The facility should take advantage of programmability frameworks and support the entire lifecycle of the Network Apps (Figure 7.17). In the lifetime of a Network App, three main processes can be defined, namely: (i) the Network App Development process, where the actual code production is performed; (ii) the Network App Testing process, where testing at various levels and for different targets is performed (including verification tests, validation tests, and certification tests); and (iii) Network App Publication process, where the process of uploading/storing a Network App to a marketplace is performed.

From a business perspective, the facility that supports those processes is meant to serve as a collaborative platform for the infrastructure owners and the vertical industries, and thus, it should engage the creation of a vertical-specific market, analogous to the currently available ones for mobile apps (e.g., Play Store and AppStore). Next, the architectural components/environments of such a facility are provided.

7.7.1 Architectural Components of the Facility

7.7.1.1 Development environment

For the facility that will support the Network App lifecycle, the adoption of a CI/CD approach enabled by DevOps software development methodology (software development – Dev and information-technology operations – Ops) is a key approach. Already, OSM and ONAP have implemented aspects of DevOps workflows to support their respective deployments. Also, there are several related tools that can enable automated lifecycle management for the Network App development process.

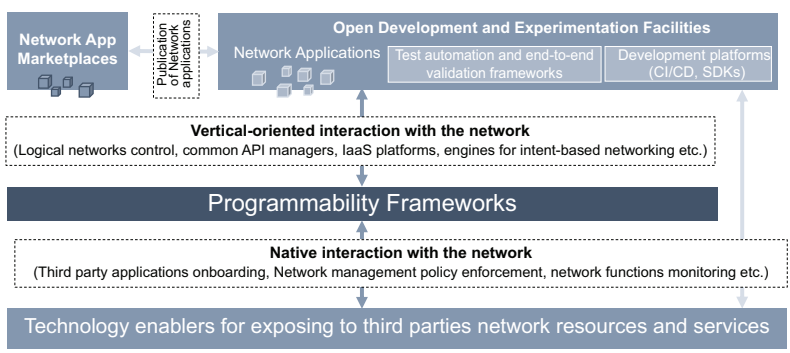

Figure 7.17. Open facility for Network App lifecycle support on top of programmability frameworks.

7.7.1.2 Validation environment

The main role of the validation environment is to provide capabilities for running automated tests under well-defined configuration/parametrization of the facility. Four main features are considered.

- **Onboarding controller:** The developments are imported into third-party servers that are part of the open experimentation facility in order to be validated and tested; thus, capabilities for controlling the onboarding process are added to the platform. To make this procedure dynamic, a set of virtual infrastructure managers and orchestrators at the facility is required.
- **Test execution manager:** It is responsible for configuring and scheduling the facility based on the target validation tests. This is performed by using the information available from the facility in conjunction with a test descriptor. The test descriptor is a structured form of information needed for conducting a test. Already, there is comprehensive work on that direction from EC Horizon 2020 5GPPP projects,[22] and 5GPPP has published a comprehensive white paper on that [39].
- **Test automation tool:** It is responsible for applying the commands from the test execution manager to the facility and to host agents or plugins required for executing a test. This is an important part of the framework since it allows the verticals to reuse a pool of tests and related plugins on demand. Thus, the design of the test and the implementation of the related agents are decoupled from the vertical service development process.
- **Results analysis and visualization:** Vertical to network-level measurement campaigns are considered in the proposed framework. The measurements can be collected from exposure APIs and directly from agents. The diversity of the available data and their volume can enable analytics. The visualization of the results is also provided, for real-time network resource monitoring and, also, for dashboarding of post-processed data.

7.7.1.3 Certification environment

As the technology evolution moves network functions to the software layer and through virtualization allows open and dynamic composition of network services extending capabilities to the business through the Network Apps concept, the established certification practice in the mobile network business needs to extend beyond the current practice and include supplementary software specification conformance and quality assessments. Network Apps, as primarily third-party software interworking with the network, shall need certification in accordance with

22. https://5g-ppp.eu/5g-ppp-phase-3-1-projects/

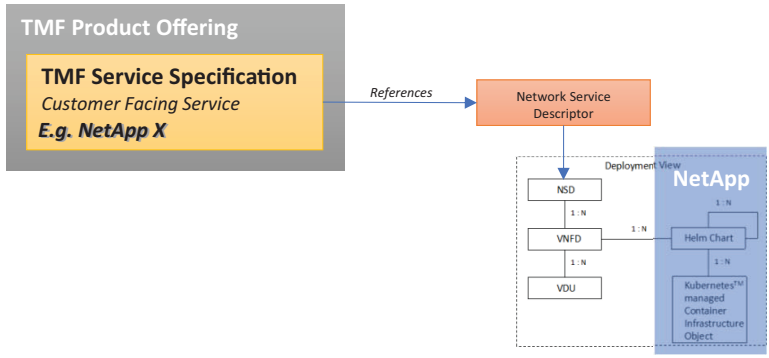

Figure 7.18. Network App exposed as TMF product offering in a marketplace.

the equipment paradigm. In that sense, the certification environment refers to the programming environment that provides all the tools and processed required for certifying Network Apps. The certification process includes testing against standards and regulations, so as to guarantee that a Network App functions properly under any scenario and data load. In contrast to the verification and validation processes where the testing is application- or scenario-oriented where the focus is on the efficiency of the Network App functionality, the certification process focuses on the correctness of the interfacing, based on the related specifications, so as to maximize the interoperability of the Network App.

7.7.1.4 Publication environment – marketplace

From the business perspective, the Network App ecosystem requires a collaborative platform for the infrastructure owners and the vertical industries, and as such, it engages the creation of a vertical-specific market, analogous to the currently available ones for mobile apps (e.g., Play Store and AppStore). Thus, the developments (Network Apps) become available to any interested party either for purchase/utilization and/or for reuse/enhance towards a new development. The role of that collaborative platform can be played by a Network App publication environment, with functionality analogous to a mobile app marketplace. Already, related 5GPPP projects develop such marketplaces, for instance, the EVOLVED-5G market place[23]; however, the usage of existing (of more general purpose) platforms (AWS marketplace, Google, etc.) as a basis is not excluded as long as the related Network App certification is guaranteed.

Another approach from the 5GASP project proposes to use the TMF's Product resource model[24] for publishing Network Apps to a Marketplace and therefore

23. https://github.com/EVOLVED-5G/marketplace

24. TMF620 – Product Catalog Management API REST Specification.

making it publicly available after the DevOps experimentation and certification readiness lifecycle is successfully completed. This approach not only incorporates adequate resources to describe a Network App offering but also ensures interoperability among other industry implementations. Taking into consideration the various properties of the aforementioned model, the highlighted ones (Figure 7.18) will be leveraged to describe a Marketplace asset. Briefly, a Marketplace asset might contain an attachment (e.g., logo, images, certification links, or files), topological information about the offered deployment, pricing, asset's specific characteristics, service level agreement (SLA) reference, and lastly, a reference to the actual services ordered and employed, i.e., hosting network slice, Network App, and test descriptor. Notable mention should be made of the latter entity, namely, Service Candidate Ref of the Product Offering resource model. This entity associates the product offering with the Network Apps Service Specifications constituting the onboarding and deployment model.

To end up, utilizing TMF's product aims at:

- Consistency between the ordering and deployment model.
- Introduction of business aspects, such as pricing, product options, and market segment.
- Imposing an abstraction layer between customers and service providers.
- Effortlessly interacting with other production systems.

7.8 Programmability Enables Intent-Based Networking

An intent is an expression of the desired state that you want to be realized and can be considered [6, 40] as portable and abstract. Portable, in the sense that it can be moved between the different controller and network implementations and remain valid; and abstract since it must not contain any details of a specific network. The advantage of an Intent is flexibility, as it allows users to express policies using concepts and terminology that are familiar to the user without having specific knowledge in the field. There are several applications where intents can be applied, including service model and orchestration ([41–44]), network orchestration ([45, 46]), monitoring and resource exposure ([47, 48]), and intent deployment and configuration [49].

To make possible the implementation of intents, without compromising the system, it is necessary to study the life cycle of an intent from its creation to its installation (see Section 7.8.1). On top of the principles that are defined from the lifecycle of the intents, programmable frameworks and middleware can be created to enable the development of related (IBN-based) Network Apps (see Section 7.8.2).

7.8.1 State Machine for IBN-enabled Industrial Networks

A reference state machine for IBN-enabled Industrial networks is depicted in Figure 7.19. This state machine is divided into six sections. After the user requests in the first phase, the validation phase verifies that the request contains all the necessary information to implement the desired action. If the user has forgotten to mention necessary information for the request to be implemented or if the information, what he has provided, is wrong, this phase assigns an invalid status, which will require user interaction. On the other hand, if the user provides all the necessary information for the request to proceed, it assigns a valid status and continues to the next phase.

The conflict phase is the second checkpoint of this state machine. Here, the already validated attempts are subject to a comparison with the requests already implemented and stored in the database to conclude conflicts. If the request is redundant, or if the new request consists of information contrary to that already implemented, the conflict phase assigns a conflict state which will be resolved with the help of the user. If the information does not conflict, the compilation section is enabled.

When the request reaches the compilation phase, the system tries to convert into rules the policies that the user wants to implement, i.e., the system converts the high-level language, from the intent, into a language that can be interpreted by the controller so that it is then possible to install them. Sometimes, the desired information may not be supported by the controller, or the desired operations may not be operational, and in this case, the compilation phase gives a compilation error that can be solved with the help of the user. If the rules to be installed are retrieved, the installation phase is enabled.

Once the rules are obtained, the next step is to install them on the controller. If the objectives are supported by the controller and installed, the monitoring phase is enabled. If the goals are not achievable because they may be offline or non-existent, the user is alerted that the requested application was not installed.

In the final phase, monitoring consists of a cycle that constantly checks if the existing requests in the database are being fulfilled or if any violation has occurred. If any non-compliance occurs, the system, through intelligent algorithms, tries to identify how to solve the problem without human intervention automatically. To make this possible, the area of ML enters this phase, whereby capturing data from the network, the system tries to identify patterns in order to validate existing policies and intervene if necessary. If the intelligent algorithms do not identify a solution that corrects the problem, the user is alerted.

This cycle of intents installation avoids problems of redundancy and inconsistency in the infrastructure, allowing the user to orchestrate networks by expressing

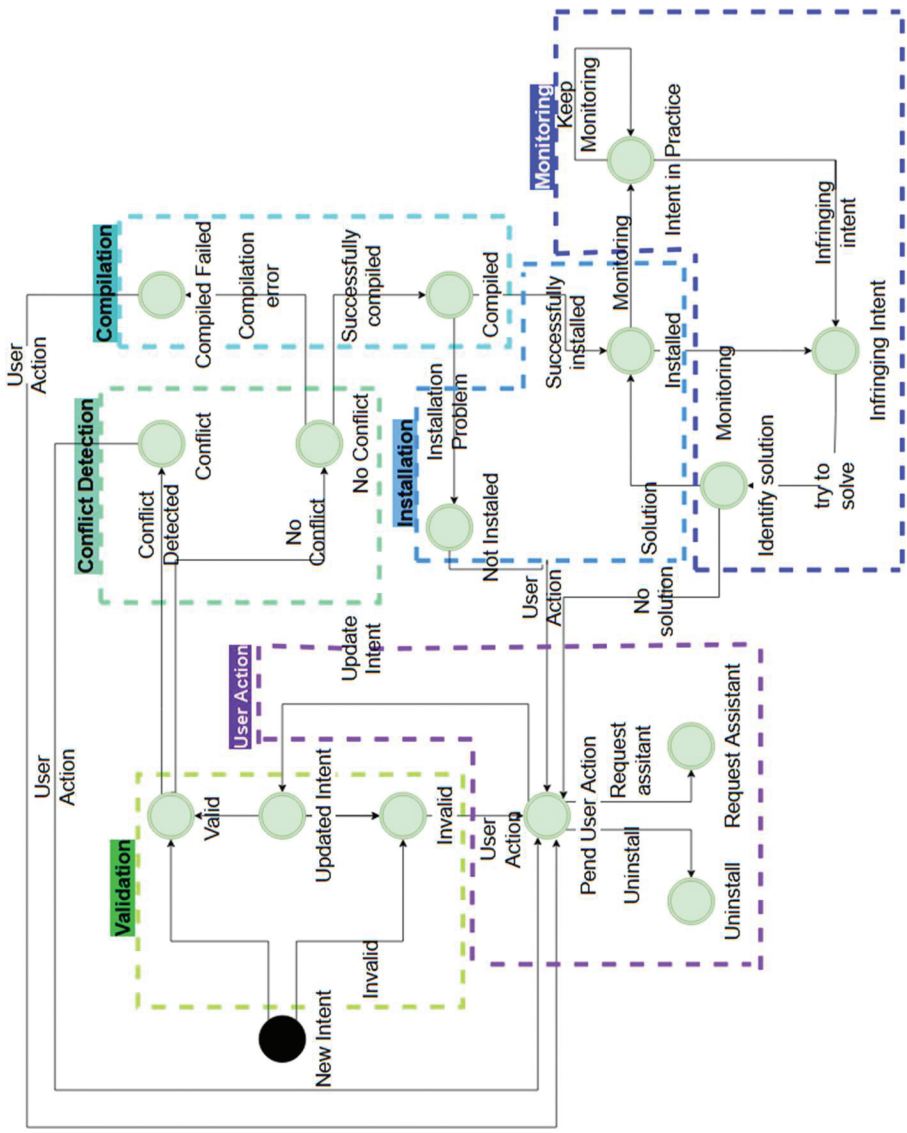

Figure 7.19 State machine for enabling IBN.

only what they want to happen, without having to worry about how it will be implemented.

Perhaps the most important application of intent-based networking in 5G industry use is automation [50], automation that is readily derived from IBN has helped to overcome the traditional massive device-to-device connection, thus ensuring great efficiency, turnaround, and scalability. With the use of a well-designed artificial model, operators are not only able to automate several processes but also provide service-level assurances.

7.8.2 Middleware for Intent-based Networking

Adopting the concept of intent-based networking, an "intent engine" is envisioned, which acts as an OSS/BSS leveraging AI/ML algorithms to automate the network application lifecycle management depending on the end users' intent, providing it to a middleware, which allows the intent engine to control the full operation of network services and network slices under its control.

The domain layer generates IBN policy based on specific domain knowledge pre-stored in a semantic model and an information model. The policy will be used by the infrastructure enablement layer to manage virtual resources.

A predefined sequence may not be compatible with the workflow of an autonomous operation. Semantic models and intent-based networking is supposed to improve service time. During the verification process, the amount of time taken to launch network service under three typical scenarios will be tested to verify the improvement of the domain knowledge on the phase.

The intent-based networking predicts the need for robots and specifies policy towards OSM and RAN controllers to deliver management, topology, placement, and resource optimization within 5G Cloud environments. An automation mechanism to align 5G orchestrators such as performance management, VNFs placement, life cycle management, and event monitoring will be implemented to reflect the intents in the optimized way. By identifying intentions and parameters autonomously specifically for the vertical domain of autonomous robots, they automate the 5G testing process accordingly.

OSM uses information model management procedure, resource, and topology. The current models do not understand the vertical users' logic behind the behaviours. A semantic model can then be used to fill the gap in between. It contains QoE models and a cookbook:

QoE models are application-specific models for translating user requirements into KPIs of QoS. The models will be derived from relevant use case patterns per application. It is essential for OSM to understand the workflow, workloads, and topology and enables the middleware to optimize service provisions for individual applications.

The cookbook contains many recipes as templates. They are prepared for vertical with multiple concurrent applications. A semantic model will create a type system to describe possible building blocks (such as a "Compute" node type, a "Network" node type, or a generic "Database" node type) of applications. They will be used in the model for constructing a behaviour template together with QoE models. The type system will then be used to define service templates (robot service, edge service, Cloud service, and collective service) which consist of lifecycle operations and behaviours of orchestration engines.

Combining QoE models and Template, user intents will be applied to derive the order of component instantiation, manage lifecycle operations, and instantiate single components at runtime with strong interpretability and interoperability.

The intent-based networking is realized by middleware with

- Semantic interpretation engine.
- Lifecycle management engine.
- Performance management engine.
- Fault management engine.
- Package management engine.
- Security manager.

An IB policy generator is the centre of the middleware, and it derives the context of the requests and then translates the request into policy through corresponding engines. The intent-based networking is supported by pre-defined receipts from the semantic database, and ML toolboxes help the engine to derive the current states of the system using events gathered. Fault, package, lifecycle, and performance management engine helps the orchestrator to define management procedures. Their policies are also specified as receipts within the rich domain model. The focus is on translating behaviours derived by the semantic interpretation engine to procedures within the information model, which is subsequently understandable by OSM and RAN Controller.

7.8.2.1 Enabling network applications for robot autonomy

In order to achieve robot autonomy, many advancements are needed beyond just another provider-centric 5G architecture or framework solely to improve quality of service (QoS). The ambition must be on the user-centric paradigm of integrating vertical knowledge into the existing 5G solutions. Under the umbrella, the software architecture proposed focuses on bridging OSM and ROS/network applications development.

The architecture is divided into four spiral layers as shown in Figure 7.20.

- The domain layer generates IBN policy from specific domain knowledge pre-stored in semantic models and information models.

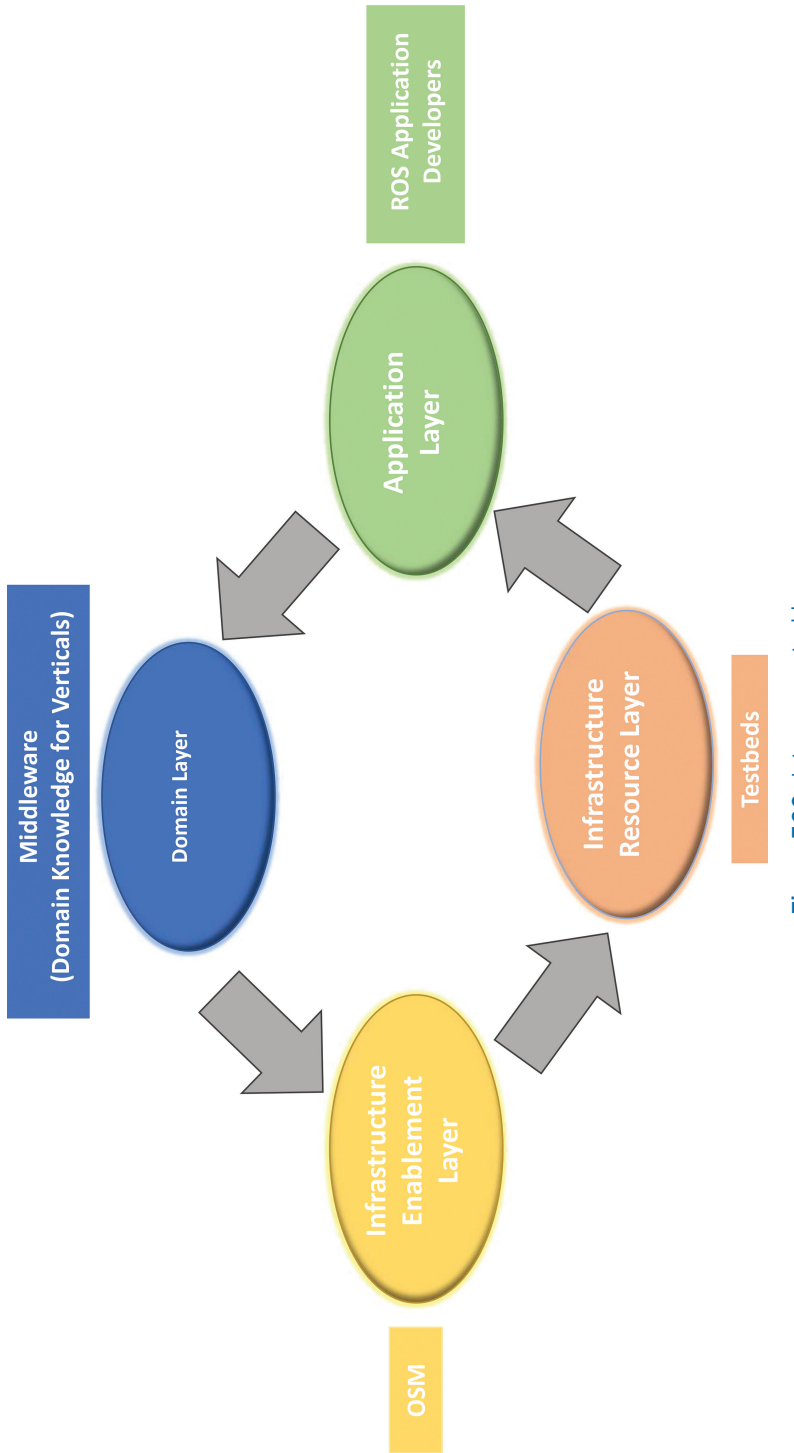

Figure 7.20 Interconnected layers.

- The infrastructure enablement layer contains ETSI OSM and RAN controller to control the core network and RAN.
- The infrastructure resource layer is closely linked to the 5G testbeds, it provides physical resources and enables network services to be deployed on Robots, Edge, and Cloud.
- The application layer delivers the network application using VNFs and KNFs. The network application is deployed to enable the ROS network for fundamental robot capabilities such as perception and user interaction. It also enables the Cloud-native service provision together with the network services.

The four layers are interconnected in a spiral shape and linked closely to the ongoing development of 5G testbeds, open-source MANO (OSM), robot operating systems (ROS), and domain-driven design. It reflects the multidisciplinary nature of the project development. The architecture enables patterns to be tangible in their specific sub-domains for further verification. The innovation leads to an automated and interpretable mechanism for deriving the placement of network functions, order of component instantiation, and effective lifecycle management. This is essential for application-driven approaches towards automatic configuration on testbeds using ML and AI and for enhanced robot autonomy in the vertical sectors.

A middleware layer of ROS-based network applications will implement common robot functions that can be invoked by the respective over-the-top (OTT) robotic applications. In this case, network applications are defined as disaggregated application enablement services, which can span across technology domains (i.e., Core and Edge).

The network applications will be implemented as VNF chains within network slice subnet instances (NSSIs), as per ETSI NFV EVE012 specifications. Thus, individual VNF instances can be optimally placed depending on resource availability, and a number of various constraints (e.g., maximum delay, maximum throughput, etc.) network applications that will implement common robotic operations such as mapping can then be shared between several robotics applications.

Thus, OTT service creation entails the instantiation of a network slice instance (NSI), which shares the network slice sub-slice instance (NSSI) resources already reserved by one or more network applications (e.g., in terms of processing power, storage, etc.), thus avoiding the costly resource reservation and VNF instantiation step and significantly reducing service creation time. Furthermore, the network applications will deliver open, standards-compliant Northbound APIs for robotics vertical applications that facilitate rapid prototyping.

The workflow that exploits NFV/SDN infrastructures for enhanced autonomy requires computing and storage to be shifted dynamically and repeatedly among robots, edges, and the central cloud. Partial information will be replicated among

network services (NSs) deployed in different locations. To tailor NSs, different configurations of VNFs and KNFs are required to achieve 5G network services. Additionally, due to limited resources, robots and edges would prefer fine-grained network functions to preserve their efficiency.

A library of generic vertical services is the centre of the reference Network Applications. They are linked to ROS simulation environments, dense learning, and model-based RL learning toolbox and optimized by specific deployment requirements of robots alone, edge along, Cloud along, or collective. The network functions of the generic library will be implemented using KNFs and VNFs and controlled by VCA of OSM under the orchestrator to the VNFs network. Network slicing of the testbeds will be customized and integrated for deploying the KNFs and VNFs in "terminal," "edge," and "remote" environments optimized deployment. The topology, placement, and life cycle management of the network functions in the reference network application will be derived based on general patterns of the 5G enhanced autonomy. To realize the Cloud-native design, generic vertical services will be implemented using Micro-services. The service definition can be obtained using the reference catalogue service. The service can be ordered and replicated using the reference order service. Internally, data consistency is ensured by CQRS and ES. Authentication is traced in the reference identity service. Overall security is controlled by the security manager (which is also part of generic vertical service). A UI can be provided for high-level monitoring of the system status and result analysis.

As an example, a new open-source library can be implemented to realize distributed map services within "terminal," "edge," and "remote" environments for a shared environment representation. Topology, placement, and the life cycle management of the networked mapping functions will be implemented in the reference network application to realize the collective intelligence required by enhanced autonomy.

Finally, the reference network application demonstrates the standardization of APIs on testing facilities. The applications within the library of generic vertical services can be developed using ROS directly. Low-level events obtained from testbeds will be propagated on the event bus and translated by a semantic interpretation engine for high-level meanings. This capability ensures interpretability. Third-party vertical developers can reuse VNFs and KNFs of the generic vertical services which have validated their compatibility from the testbeds. Therefore, the experimental facilities are exposed to the developer. They can be expanded for use case-specific functions such as 5G enhanced perception, detection, and planning in vertical sectors such as PPDR and healthcare, transport, and industrial 4.0. A key innovation, namely, Cloud-native applications for NSs and standard APIs can be realized by reference network applications and ROS.

7.9 Conclusions

The next generation of mobile networks will take full advantage of the convergence between the IT and telecom sectors. Through this convergence, 5G has already transformed the mobile network infrastructure to a flexible service provisioning platform, and 6G is expected to provide mobile networks as fully programmable platforms, with native cloud capabilities at any network domain and communication plane. This chapter describes current technology enablers that contribute towards a fully programmable mobile network, by clustering them to those at the deployment and connectivity level, at the network and resource management level, and at the service and application provisioning level. To exploit those enablers, ongoing research and standardization work is being conducted, with the main target being the provisioning of programmability frameworks, i.e., frameworks, that abstract the network underlay infrastructure and its capabilities so that they are dynamically controlled and configurable. Some indicative approaches are described in this chapter, including the deployment of common API managers, the exploitation of P4-programmable switches, the usage of open interfaces of O-RAN, and the design of SDKs for providing network slices as a service. The potential of programmable networks is high and yet to be investigated in detail. However, as it has been indicated already, concepts such as intent-based networking take advantage of network programmability features. Overall, new business models emerge since, through programmability, third parties can develop and integrate their solutions (e.g., their network applications) into the underlay connectivity and compute infrastructure.

References

[1] D. Tsolkas and H. Koumaras, "On the Development and Provisioning of Vertical Applications in the Beyond 5G Era," in *IEEE Networking Letters*, vol. 4, no. 1, pp. 43–47, March 2022, doi: 10.1109/LNET.2022.3142088.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and. Walker, "P4: Programming protocol-independent packet processors," In *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[3] M. Boucadair and C. Jacquenet, "Introducing Automation in Service Delivery Procedures: An Overview.," in *Handbook of Research on redesigning the future of internet architectures*, Hershey, PA, USA: Information Science Reference, an imprint of IGI Global, 2015.

[4] A. Aijaz, "Private 5G: The Future of Industrial Wireless," In *IEEE Industrial Electronics Magazine*, vol. 14, no. 4, pp. 136–145, Dec. 2020, doi: 10.1109/MIE.2020.3004975.

[5] Y. Wei, M. Peng, Y. Liu, "Intent-based networks for 6G: Insights and challenges," In *Digit. Commun. Networks*, vol. 6, pp. 270–280, 2020.

[6] A. K. Salkintzis, "Interworking techniques and architectures for WLAN/3G integration toward 4G mobile data networks," In *IEEE Wireless Communications*, vol. 11, no. 3, pp. 50–61, 2004.

[7] *Summary of rel-16 work items*, Technical Report (TR) 21.916, v16.0.0, 3GPP, June 2021. Accessed: April 6, 2021. [Online] Available: https://www.3gpp.org/ftp/Specs/archive/21_series/21.916/.

[8] *Study on Communication for Automation in Vertical Domains (Release 16)*, technical Report (TR) 22.804, v16.2.0 3GPP, Dec 2018.

[9] *System architecture for the 5G System (5GS); Stage 2 (Release 17),* Technical Specification (TS) 23.501 v17.5.0, 3GPP, June 2022.

[10] M. Corici, E. Troudt, P. Chakraborty, and T. Magedanz, "An Ultra-Flexible Software Architecture Concept for 6G Core Networks," In *2021 IEEE 4th 5G World Forum (5GWF)*, pp. 400–405, 2021, doi: 10.1109/5GWF52925.2021.00077.

[11] M. Corici, E. Troudt, and T. Magedanz, "An Organic 6G Core Network Architecture," In *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, pp. 1–7, 2022, doi: 10.1109/ICIN53892.2022.9758088.

[12] M. Corici, E. Troudt, T. Magedanz, and H. Schotten, "Organic 6G Networks: Decomplexification of Software-based Core Networks," In *2022 Joint European Conference on Networks and Communications (EUCNC) & 6G Summit*, pp. 541–546, 2022, doi: 10.1109/EuCNC/6GSummit54941.2022.9815730.

[13] *O-RAN Use Cases Analysis Report*, Technical Report v10.00 Rel. 003, O-RAN WG1, March 2023.

[14] 5G-PPP Software Network Working Group Network Applications: Opening up 5G and beyond networks 5G-PPP projects analysis September 2022, DOI: 10.5281/zenodo.7123919.

[15] N. Foster, J. McKeown, G. Rexford, L. Parulkar, Peterson, and O. Sunay, "Using deep programmability to put network owners in control" In ACM SIGCOMM Computer Communication Review, vol. 50, no. 4, pp. 82–88, Oct. 2020, doi: https://doi.org/10.1145/3431832.3431842.

[16] TM Forum, "The Open API project," 2023. Accessed: April 6, 2023. [Online]. Available: https://www.tmforum.org/collaboration/open-api-project/.

[17] D. Fragkos, G. Makropoulos, P. Sarantos, H. Koumaras, A. -S. Charismiadis, and D. Tsolkas, "5G Vertical Application Enablers Implementation Challenges and Perspectives," In *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pp. 117–122, 2021, doi: 10.1109/MeditCom49071.2021.9647460.

[18] *Functional architecture and information flows to support Common API Framework for 3GPP Northbound APIs; Stage 2, (Release 17)*, Technical Specification (TS) 23.222, v17.6.0, June 2022.

[19] 5G-IA, "EU vision on 6G," Whitepaper, June 2021. Accessed: April 6, 2023. [Online]. Available: https://5g-ppp.eu/european-vision-for-the-6g-network-ecosystem/.

[20] ETSI "ETSI TeraFlow SDN (TFS)," 2023. Accessed: April 6, 2023. [Online]. Available: https://tfs.etsi.org/.

[21] H. Lønsethagen, S. Lange, T. Zinner, H. Øverby, L. M. Contreras, N. Ciulli, E. Dotaro, "Towards Smart Public Interconnected Networks and Services – Approaching the Stumbling Blocks," Preprint in *TechRxiv*, 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.36227/techrxiv.19690570.v1.

[22] IETF, "Framework for IETF Network Slices," 2023. Accessed: April 6, 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slices/.

[23] IETF, "Challenges for the Internet Routing Infrastructure Introduced by Changes in Address Semantics," 2023. Accessed: April 6, 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-king-irtf-challenges-in-routing/.

[24] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: an SDK for next-generation SD-RANS," In *Proc. 17th International Conference on Emerging Networking EXperiments and Technologies (CONEXT 2021)*, 7–10 December 2021, Munich, Germany (Virtual Conference).

[25] Linux Foundation, "Traffic Control tc (8), Linux man page," 2023. Accessed: April 6, 2023. [Online]. Available: https://linux.die.net/man/8/tc.

[26] Linux Foundation, "Open vSwitch," 2023. Accessed: April 6, 2023. [Online]. Available: https://www.openvswitch.org/.

[27] NetFPGA, "About NetFPGA," 2023. Accessed: April 6, 2023. [Online]. Available: https://netfpga.org/About.html.

[28] Netronome, "About Agilio SmartNICs," 2023. Accessed: April 6, 2023. [Online]. Available: https://www.netronome.com/products/smartnic/overview/.

[29] R. Ricart-Sanchez, P. Malagón, A. M. Escolar, J. M. Alcaraz Calero, and Q. Wang "Toward hardware-accelerated QoS-aware 5G network slicing based

on data plane programmability," In *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 4, January 2020.

[30] 6G BRAINS, "D5.1 E2E network slicing control enablers," December 2021. Accessed: April 6, 2023. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5f560ab13&appId=PPGMS.

[31] H. Harkous, M. Jarschel, M. He, R. Pries, and W. Kellerer, "P8: P4 with predictable packet processing performance," In *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2846–2859, 2021.

[32] H. Harkous, B. A. Hosn, M. He, M. Jarschel, R. Pries, and W. Kellerer, "Towards performance-aware management of p4-based cloud environments," In *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV- SDN)*, pp. 87–90, 2021.

[33] *NR and NG-RAN Overall description, Release 17*, Technical Specification (TS) 38.300, v17.2.0, 3GPP, September 2022.

[34] ETSI, Common API Framework for 3GPP Northbound APIs (3GPP TS 23.222 version 16.8.0 Release 16) TS 123 222 V16.8.0 (2020-10).

[35] *Security aspects of Common API Framework (CAPIF) for 3GPP northbound APIs (Release 17)*, Technical Specification (TS) 33.12, v17.0.0, 3GPP, March 2022.

[36] A. M. Sanchez, A.-S. Charismiadis, D. Tsolkas, D. Artuñedo Guillen, and J. G. Rodrigo" Offering the 3GPP Common API Framework as Microservice to Vertical Industries," In *EuCNC & 6G Summit* 2022, June 2022.

[37] *Common API Framework for 3GPP Northbound APIs; (Release 17)*, Technical Specification (TS) 29.222, v18.0.0, December 2022.

[38] D. Fragkos, G. Makropoulos, A. Gogos, H. Koumaras, and A. Kaloxylos, "NEFSim: An open experimentation framework utilizing 3GPP's exposure services," In *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, June 2022.

[39] L. Nielsen, A. F. Cattoni, Z. Diaz, G. Almudena, B. García, A. Gavras, M. Dieudonné, and E. Kosmatos, "Basic Testing Guide - A Starter Kit for Basic 5G KPIs Verification," 5G PPP Whitepaper, 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.5704519.

[40] *Experiential Networked Intelligence (ENI); Context-Aware Policy Management Gap Analysis*, ETSI GR ENI 003, v1.1.1, March 2018. Accessed: April 6, 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/ENI/001_099/003/01.01.01_60/.

[41] A. Rafiq, A. Mehmood, and W. C. Song, "Intent-based slicing between containers in sdn overlay network," In *J. Commun*. Vol. 15, no. 3, March 2020.

[42] F. Paganelli, F. Paradiso, M. Gherardelli, and G. Galletti, "Network service description model for vnf orchestration leveraging intent-based sdn interfaces," In *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017.

[43] W. Cerroni, C. Buratti, S. Cerboni, G. Davoli, C. Contoli, F. Foresta, F. Callegati, and R. Verdone, "Intent-based management and orchestration of heterogeneous open-flow/iot sdn domains," In *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017.

[44] A. Rafiq, A. Mehmood, T. A. Khan, K. Abbas, M. Afaq, and W. C. Song, "Intent-based end-to-end network service orchestration system for multi-platforms," In *Sustainability*, vol. 12, no. 7, 2020.

[45] K. Abbas, M. Afaq, T. A. Khan, A. Rafiq, and W. C. Song, "Slicing the core network and radio access network domains through intent-based networking for 5G networks," In *Electronics*, vol. 9 no. 10, pp. 1710, 2020.

[46] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, H. Flinck, and M. Namane, "Benchmarking the ONOS intent interfaces to ease 5g service management," In *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018.

[47] Y. Tsuzaki and Y. Okabe, "Reactive configuration updating for intent-based networking," In *2017 International Conference on Information Networking (ICOIN)*, 2017.

[48] A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, "Refining network intents for self-driving networks," In *Proceedings of the Afternoon Workshop on Self-Driving Networks, SelfDN*, pp. 15–21, 2018.

[49] F. Aklamanu, S. Randriamasy, E. Renault, I. Latif, and A. Hebbar, "Intent-based real-time 5G cloud service provisioning," In *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018.

[50] S. Garg, M. Guizani, Y. -C. Liang, F. Granelli, N. Prasad, and R. R. V. Prasad, "Guest Editorial Special Issue on Intent-Based Networking for 5G-Envisioned Internet of Connected Vehicles," In *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5009–5017, Aug. 2021, doi: 10.1109/TITS.2021.3101259.