Cloud-Native 5G Service Delivery Platform

Kay Haensge
InterDigital Germany GmbH
Berlin, Germany
kay.hansge@interdigital.com

Dirk Trossen, Sebastian Robitzsch
InterDigital Europe Ltd.
London, UK
{dirk.trossen,sebastian.robitzsch}
@interdigital.com

Michael Boniface, Stephen Phillips
IT Innovation Centre
Southampton, UK
{mjb,scp}
@it-innovation.soton.ac.uk

***Abstract*—Operators have been adopting the cloud-native paradigm for the rollout of 5G architecture. With it goes the introduction of the service-based architecture as a key design pattern for realizing future control, and ultimately user planes of mobile networks. The main benefits of this new design pattern are the increased flexibility to address new business cases while maintaining competitive cost levels, while also benefitting the realization of use cases usually requiring the full breadth of infrastructure level network slicing. In this paper, we present a realization of a service delivery platform entirely built upon service-based architecture principles, showcasing the realization of control as well as user plane services along with early deployment insights.**

Keywords—service-based architecture, service routing

I. Introduction

The 5G community is abuzz with excitement about the introduction of the 'cloud-native' paradigm. It is seen as one of the most important departures from 4G in terms of deploying future (5G) mobile networks, alongside the ability to slice networking and compute resources; in fact, many see it as a basis for an efficient slicing realization in the first place.

A 'cloud-native' application is a program designed specifically for a cloud computing architecture. It leverages the benefits of cloud computing frameworks that are composed of loosely coupled cloud services. They utilize design patterns and deployment techniques in today's (Internet) cloud services. This affects two crucial aspects, namely the *relationships* in the overall system as well as the *design of services* realized in a cloud-native system. For the former, the assumptions for what constitutes a 'cloud' and the abstraction provided to upper service platforms is crucial, while the latter aspect drives not only the realization of key 5G use cases but also the assumption of *basic platform capabilities*. Current work in the NGMN forum and the accompanying efforts in the 3GPP standards community address both, with the most recent Rel16 specification capturing the realization of 5G control planes [1].

In this paper, we will build on the cloud-native assumption of 5G and present our efforts to realizing a service platform, not limited to the realization of control plane services but also being deployed with user plane services, specifically those for media consumption, in mind. This realization is captured as one of three possible deployment choices for 5G cloud-native service platforms in (Appendix G4) [1]. For this, we will first outline the key relationships in a cloud-native 5G deployment in Section II, with our approach to realizing the service delivery platform in Section III. We present our approach to lifecycle management in Section IV before outlining first insights into performance in Section V, including those based on real-life deployments of our platform.

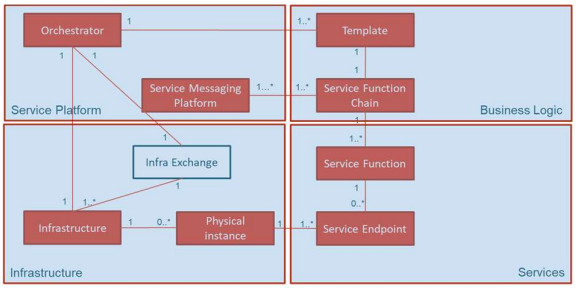

Figure 1: Service Interaction Pattern

II. Relationships & Interactions in Cloud-Native Environments

Figure 2 outlines the relationships in the cloud-native deployment of services, following the well-established model found in data centre deployments [2][3][4]. At the very bottom, the infrastructure is being provided, e.g., by wholesale communication providers such as operators or facility providers in vertical 5G scenarios. The assumptions for the infrastructure here follow that of data centres, i.e., compute resources are provided via a Layer 2 connectivity between clusters of such compute resources, ultimately providing a data centre abstraction to the upper most services in which services can be deployed as service instances, using virtualization solutions such as containers or virtual machines. These data centre resources now exhibit a higher degree of distribution, often denoted in tiers of deployment that reach from central clouds over metro-level (central) offices to possibly even street-level deployments, e.g., in base stations or smart city furniture.

The glue between the virtualized services and the (cloud-native) infrastructure is the service messaging platform (SMP), shown in Figure 2. It is this SMP that is the focus of ongoing 3GPP work, entitled (enhanced) service-based architecture [1][5]. Section III will present the realization of precisely this platform.

Before we do so, however, we will discuss in more detail how to realize the service design patterns for such cloud-native environments. For this, Figure 2 expands our relationships in Figure 1 towards concepts that allow for capturing the interactions within the system, driving the realization of our platform in Section III.

For the management as well as control of service deployment, the business logic is captured in the form of a

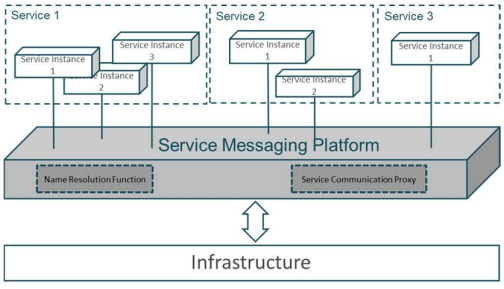

Figure 2: Relationships in a Cloud-Native 5G Environment

template as a set of instructions. Those instructions are formulated along a *service function chain* (SFC), which in turn defines an ordered set of abstract *service functions* (SF) and ordering constraints that must be applied to packets and/or flows selected as a result of classification [6]. Service functions communicate through the *service messaging platform*, in alignment with Figure 2.

The service functions themselves realize the service (logic) and can exist as several *service function endpoints* (SFEs) in the distributed cloud-native infrastructure. The latter is realized with the help of the *infrastructure* providing one or more *physical instances* for such service endpoint.

Services running on the platform rely on a lifecycle model, which is managed by the (platform) *orchestrator*, translating the instructions in the provided template to initially deploy suitable physical instances of the defined service endpoints within the infrastructure resources. For this, the service platform and infrastructure player maintain an agreed resource pool, which underlies a well-defined business relation (usually expressed through a quota of resource usage). Beyond the initial deployment, the orchestrator also controls the runtime behaviour of the physical instances, said behaviour being defined as part of the initially provided template and changes triggered through monitoring and analysis of state and behaviour across the full stack of service function chain, service endpoint and infrastructure realisation

III. Service Platform

Figure 3 depicts the overall service platform architecture, embedded into the relationships already shown in Figure 2 [7]. At the topmost level, services such as those needed for 5G control planes [1] or for media services are realizing the service functions outlined in Figure 1, along the defined templates for deployment and utilizing existing Internet protocols such as HTTP for service communication.

The underlying infrastructure is providing resources to the service platform within a single network slice (the mechanisms for establishing said slices are left out of this paper), using OpenStack for the compute orchestration and OpenFlow for flexibly configuring the communication between the networking elements (here SDN switches) [8][9]. The platform itself is working in tenant mode and therefore respects possible other tenants of the infrastructure. OpenStack deploys the presented components of the platform as well as cluster runtimes on different locations including those that can be geographically distributed, which are then used by the platform to deploy Service Function Endpoints. Between the clusters and all deployed platform components, SDN-based switches perform the establishment and forwarding of packets/frames.

With this, the platform utilizes standardized interfaces and protocols both south- and northbound. In the following sections, we provide more detail on the workings of the platform components.

A. Platform Orchestrator

Orchestration is the automated process for deploying compute, storage and networking elements. It models the requests of the stakeholders into a manageable (cloud) infrastructure and performs deployment and work plans according to the requirements of theses business partners and their applications. The platform orchestrator offers interfaces to the service components and their providers as well as to the Cross-Layer Management and Control (CLMC), the latter providing a big data analysis component offering access to the multi-site cloud environment.

The orchestrator fully abstracts the infrastructure; hence service providers do not directly interact with the infrastructure components. They express their deployable SFC and requirements through templates based on TOSCA (Topology and Orchestration Specification for Cloud Applications). It provides a type system describing the blocks for constructing service templates and offering possibility to define virtual instance templates. Characteristics of these nodes can be defined within the template: properties, relationships, hardware and software requirements, policies, machine status, and deployment plans. The general idea is to define all the topics (e.g., protocols, equipment, environment variables) related with the network virtualization [10].

B. Service Function Endpoint Management and Control (SFEMC)

The SFEMC processes the computational instance requests from the Orchestrator and maintains lifecycle states of the deployed endpoints. It decides, based on given policies whether and on which infrastructure location (namely host or cluster) the described instances may be deployed. When triggered from external sources (e.g., CLMC or Service Providers), the SFEMC performs targeted transitions of the deployed lifecycle state of each SFE and applies given policies against it. As the maintainer of these lifecycles, the SFEMC retrieves monitoring status information from the CLMC as well as from the underlying virtual instance managers (VIM) of the different clusters within the platform.

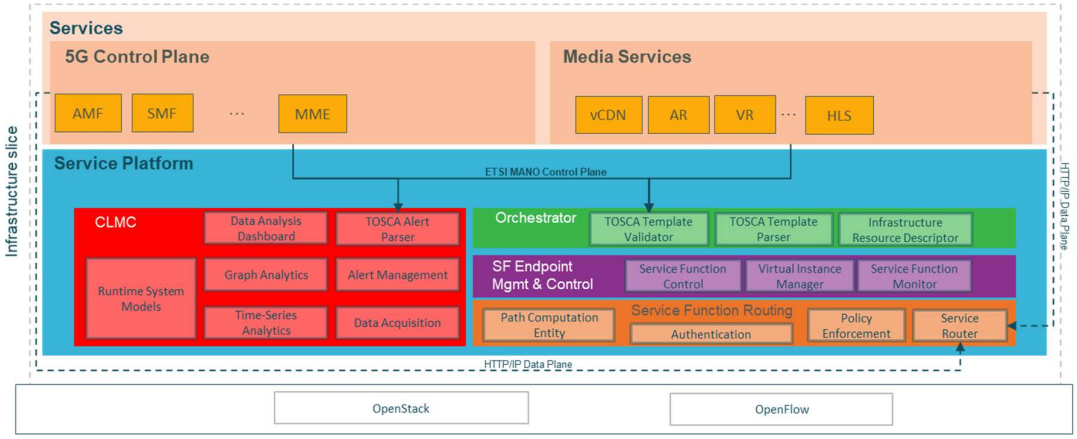

Figure 3: Platform Architecture

C. Service Function Routing

Service Function Routing (SFR) enables service-oriented routing of traffic among SFEs based on HTTP as the application protocol via SDN-enabled Layer 2 forwarding. Through the registration of the FQDN (Fully Qualified Domain Name) of the specific service, an SFE is serving as an endpoint within our relations in Figure 1. Our platform utilizes so-called *name-based routing* for the delivery of IP-based services in a Layer 2 network, as described in [11].

D. Cross Layer Management and Control (CLMC) for Dynamic Adaption

CLMC provides monitoring, analysis and control for adaptation of service function chains in response to usage, performance and service objectives. CLMC data can be used for control-level decisions (e.g., the Alert Management is triggering a lifecycle change of service endpoints) but also as a rich pool of data to develop insights into resource specifications, adjusting crucial longer-term strategies (e.g., placement or dimensioning) and monitoring of expected Service Level Agreements (SLAs) in B2B and B2C relationships. CLMC brings together time-series and graph analytics to understand demand, resourcing and performance properties of service function. For a given orchestration, the infrastructure and service function nodes remain largely static whilst the deployment and state of endpoints varies throughout the lifecycle of the service function chain according to demand and policies. The time-series measurements are acquired from existing monitoring frameworks provided by individual sub-systems (e.g., switching elements) or by service functions themselves. The aim is not to replace existing systems but provide a cross-layer knowledge model that can drive QoS-orientated of both content and service configuration.

IV. Lifecycle Management

In this section, we expand on the lifecycle management, as realized by the platform orchestrator, SFEMC and CLMC components presented in the previous section. For this, we differentiate the management from the control tasks, as outlined in the next sub-section. But before we do so, we describe the lifecycle state that is being managed and controlled.

A. Maintaining Lifecycle State

For each Service Function Endpoint, the platform maintains a runtime lifecycle in the form of a state machine, depicted Figure 4, with the following states:

- NON-PLACED: the SFE is known to the network but there is no placement on any cluster. However, the repository contains the container or VM image for that SF.
- PLACED: the SFE is occupying disk space on the targeted cluster but is shut down.
- BOOTED: the SFE is up and running but is not attached to the communication process. A discovery of such SFs will never return the "just" booted SFEs.
- CONNECTED: the SF Endpoint is fully in operation mode, i.e., allows communication and is discoverable.

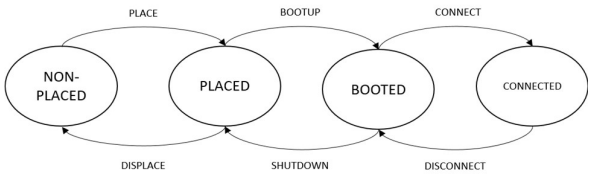

Figure 4: Lifecycle State Machine of an SFE

Hence, through the above states, the resource consumption of each SFE can be controlled across the dimensions of storage, compute and communication resources. As we will describe in the next two sub-sections, the SFE state is set up as a result of the management actions, while being controlled over time via policies that are defined in the deployment instructions.

B. Management

1) Service Function Decomposition and Packaging

Initially a Service Function Chain is based on micro services decomposed into service elements, ensuring fine granularity in service design needed to allow for service functions to be intelligently deployed on the most appropriate computing resource (e.g., geographically). Each service function is packaged as a container or VM image including installation and setup of all required software and platform specific management services. Finally, the service function packages are published into a repository accessible to the platform and supporting runtime discovery during service provisioning.

2) Service Function Chain Specification

The SFC is specified with a TOSCA template. The Platform works with a more specified *node* and *policy* type definition to leverage the platform's features. The template is labelled with service function chain identifier. All SFs are described as nodes within the SFC with named identifiers (FQDN) to be addressed when the instances of that SF are deployed. Via the policies, the specification defines how, and which SFEs will be deployed and on which location (cluster) and in which lifecycle state.

3) Orchestration

The orchestration instantiates an SFC through the deployment of SFs according to the TOSCA template specification. The specification is parsed for syntactic and semantic validation so the SFC can be processed to manageable logical objects, which in turn are forwarded to the SFEMC for the actual endpoint placement and control. Resources are then allocated via the given policies defined in the template, i.e., the SFEMC performs a validity check of possible resource requirement and places the listed Service Functions accordingly on each of the named clusters. Service Function packages are distributed to each cluster from the platform repository where they are cached. Finally, each Service Function Endpoint is instantiated to the lifecycle state within the initial policy.

C. Control

1) Bootstrapping of SFE with platform information

When an SF Endpoint enters the CONNECTED state, the endpoint's *WhoAmI*-service receives relevant platform and endpoint information. The information is available in the instance's runtime environment. Each SF has a *WhoAmI*-service installed to discover runtime context from the platform; including the Service Function Chain, Service Function Chain Instance, Cluster, Service Function IDs, and Endpoint IDs. Much of this data is automatically used by the monitoring agent to contextualise the monitoring data to feed the CLMC. For the service developer, the *WhoAmI*-API is of importance if there are multiple service functions in the service function chain; as the information is used by one SF to know the FQDN of another SF to communicate.

2) Monitoring the SFE

Alert trigger events are defined to indicate certain states of the deployed service and/or chain. Each alert specification includes an *eventType* that refers to the process used to create the alert. This includes how data is processed and the conditions under which the alert is triggered, for example, *threshold*, *relative* or *deadman* conditions. A set of configuration defines the conditions for triggering an alert that includes the metric and threshold for the critical value compared to the measured. If the comparable value hits the *threshold*, an alert is triggered. A *relative* event type is an alert that computes the difference between the current aggregated value of a metric and the aggregated value reported a given period ago. Finally, the *deadman* event type can be configured to alert if there are less points been reported in the given period as expected.

The time series analytics are extended towards the concept of temporal graphs allowing system properties to be analysed through network-aware topological structures of infrastructure, endpoints, service functions and service function chains. Graphs are created dynamically to support specific analytics and queries over the system properties. However, the model of the infrastructure properties is consistent. The properties of specific SF Endpoints vary depending on the type of service, although the common KPI taxonomy allows for general abstractions, aggregation and normalisations can be defined (e.g., Response Time).

Figure 5 shows the process of graph building and analytics. The process is initialised to build the initial graph from the infrastructure and media service topologies, along with configuring the continuous queries to acquire, aggregate and normalise the desired SF Endpoint properties over a specified time-period. The continuous queries execute periodically to add new nodes to the graph representing state of a SF Endpoint over the period. The graphs are created automatically from the measurement context data reported by the monitoring agents. A subsequent graph query is executed continuously to determine system measurements and stores these as monitoring data in time-series database for visualisation or further higher-level analytics and event triggers to control SFEs via the SFEMC.

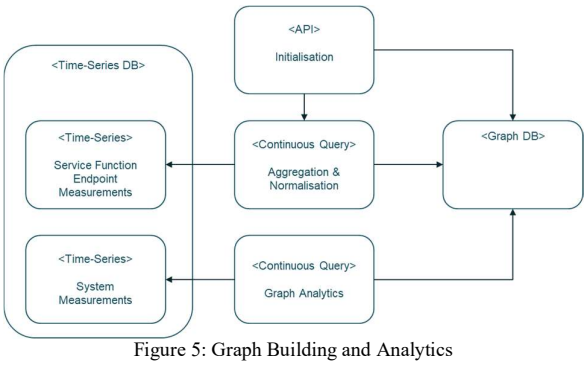

Figure 5: Graph Building and Analytics

V. Deployment Insights

In the following, we want to present use-cases which rely with diverse requirements on the platform.

A. Dev-Ops Pipeline for Experimenters

To allow a fast adoption of the key concepts of the platform, a multi-stage pipeline has been developed which allows application developers to develop and test their SFC at each stage of the development. The Dev-Ops tool chain depicted in Figure 6 fosters a smooth transitioning of the technology readiness early technology development to technology demonstration in a suitable environment.

The development and packaging of service functions is conducted outside of any platform deployment. Once the SFs have been finished (and tested on localhost) they can be packaged up to be given to the orchestrater of any deployed platform. The next step is the sandbucket. It is a Virtual Appliance containing the platform (without CLMC) to be deployed in the experimenter's infrastructure using VIMs such as VirtualBox or VMware. The objective of the sandbucket is to allow experimenters to test TOSCA templates and their SFC with the focus on the interaction among SFs.

1) Sandpit

The Sandpit is a fully virtualised environment, offering a one-tier-edge-computing-platform with four locations to test entire SFCs at a larger scale. Each location allows experimenters to deploy SFs and test the entire chain against the platform using one emulated UE per location. In contrast to the sandbucket, the sandpit also includes the CLMC and sufficient capacity to test resource and alert descriptors.

2) Replication Sites

Once the SFC has been successfully tested in the sandpit experimenters move to the city deployment of the platform which offers the abilities to conduct KPI-driven experiments and trials with members of the public. The replication sites to date are:

- Bristol: A hardware-based SDN switching fabric utilising the cities fibre-network to interconnect two edge locations in the city and the main data centre in

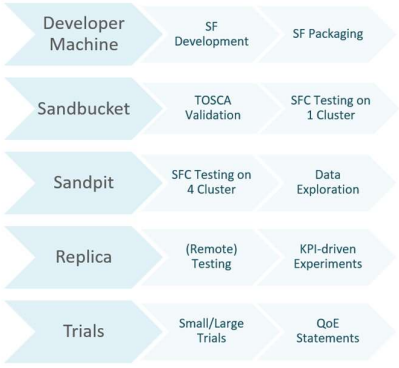

Figure 6: Pipeline to Deploy A Service Function Chain

the University. Experimenters have eight locations in total across the city where SFs can be deployed right at the far edge of the network covering attractive public areas ideal for conducting trials.

- Barcelona: A deployment at the heart of the city with four locations along a street allowing experimenters to design services including vehicular terminals. The street deployment is complemented by a data centre at the University interconnected by a fibre link.

Only with trials and KPI-driven experiments in cities experimenters can provide an educated answer to key benefits of the service-based platform, the deployment and of SFs at the edge. This requires a thorough but agile enough pipeline for designing, implementing, testing and verifying SFCs over an innovative service-based delivery platform where the pipeline controls testing costs at each stage of development.

B. Observed Platform Benefits

1) Deploying Edge-Services

From a network perspective, services are deployed as close as possible to the client (excluding the terminal perspective) where proximity is defined as the number of hops from the client to the endpoint. Through the forwarding of user packets via shortest path, the deployed edge service is used as soon as the instance is available, instead of a more centralised instance. The benefits include network cost reduction due to lower network usage as service/content is front loaded to the edge (whilst maintaining reachability from anywhere in the network), reduction in start-up time for playout on user terminals for, e.g., media services, and better reaction time due to low-latency were latency stays comparable even when client moves. These benefits, however, need to be balanced against the compute and storage usage costs for placing SFE in edge data centres.

2) Multicast Delivery of HTTP Responses

In [11], the authors outline the opportunistic multicast delivery of content, e.g., for media services. Here responses to quasi-concurrent (HTTP) requests are delivered using in a single lightweight multicast transmission over the L2 customer network. The time period (which we call the catchment interval) over which this process takes place can be flexibly adapted on a per-service request basis, further improving the opportunity for multicast delivery. The benefits include network cost reduction due to reduced network usage as video is (partially) multicast in segments of the network. Furthermore, service costs are reduced due to reduced server load as http request suppression when multicast occurs. The work in [11] outlines the possible reductions for typical customer access networks of national scale.

3) Service Indirection

The approach to service routing, utilized in our platform and based on the methods described in [11], also lowers latencies experiences in service indirection, i.e., the directing of service request to the most suitable service instance. In virtualized environments, such as the ones envisioned in edge computing deployments and 5G control planes, such service instances can be flexibly deployed through a lifecycle management system, like the one employed in our platform (see Section IV). Unlike approaches used in content delivery networks (CDNs) that realize indirection through appropriate DNS record configurations, the solutions in our platform utilize publish-subscribe name registration (and therefore also update) mechanisms. Through this, changes in service routing, i.e., the indirection to a different service instance, can be kept as low as 1ms in medium sized networks such as those found at the edge of the network (including mobile networks), while initial service lookups are comparable to initial DNS requests for very fast private DNS resolver solution (in the order of 10ms for a lookup) [11].

4) Adaptive Resource Management

The business logic for placement and scaling is described through alerts and provide an event-based approach change the lifecycle state of a service function endpoints in response to changes in performance or usage. These features allow scaling, e.g. “scale up and down” service instances for the needs of traditional cloud applications, but also the geographical scaling across a possible distribution of service instance in different data centres. The benefits here include rapid response to changes of demand and load balancing especially the spatial and temporal distribution of requests needed to support optimal resourcing of localised services

C. Example Deployments

1) Control Plane Services for 5G Core Networks

As an example, deployment for the realization of 5G control plane services, Control Plane Services (CPS) were developed to show how an upcoming 5G core network can utilize the service-based design and the capabilities of a delivery platform, most notably service registration & discovery, routing of services messages and failover mechanisms.

For testing the core network aspects only, the access network is left out and the initial requests are initiated from a virtual terminal. Upon start of such terminal, it performs an attachment to a network, followed by three user plane Sessions (for enhanced mobile broadband, video, VoLTE), finally detaching from the network again. For the realization of this simple control plane scenario, each CPS SFE can communicate with any other CPS SFE utilising the service routing capabilities of the platform for HTTP services.

We also realized stateless CPS instances by maintaining a network data service (NDS) for each service request. For instance, the session management CPS checks for permission from the NDS before requesting the respective policies from the policy management CPS. The NDS is also used for visualizing the transactions via a web interface, showing the dynamic forwarding of requests to the always nearest service function which were available. All CPS instances were set to the CONNECTED state upon deploying, not utilizing the dynamic lifecycle management outlined in Section IV.C.

We demonstrated this approach to implementing a fully SBA-compliant 5G core network at the recent NGMN (Next Generation Mobile Networks) exhibition in 2019 with Figure 7 showing the aforementioned web interface visualizing the transactions.

2) Enhanced Slicing

As mentioned in Section III, our platform is deployed within a single network slice within which services, e.g., control plane or media services, are being deployed. This choice is mainly driven by our specific deployments and their support for network slicing.

While we have considered deployments that span several network slices, we have more specifically worked on extensions to network slicing through a concept called ‘super slice’. Here, resources are being pooled (and managed) within

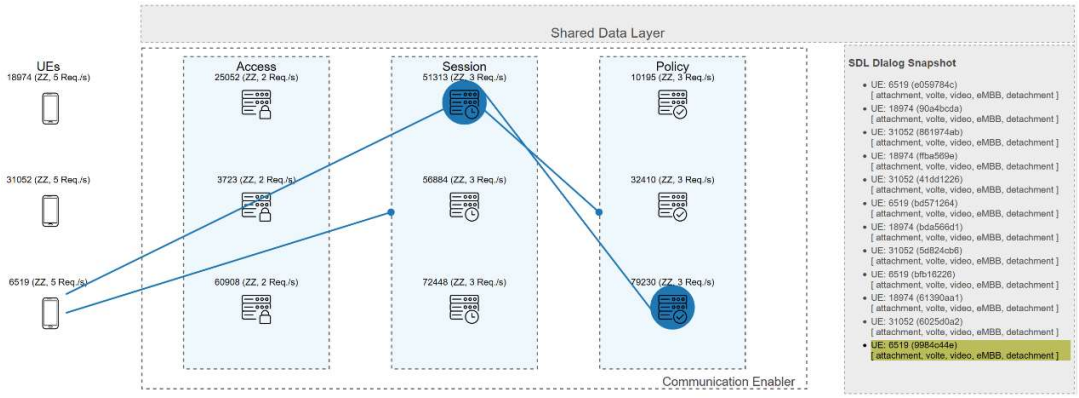

Figure 7: Control Plane Service Signaling WebGUI

a single slice. We foresee such slice being associated, by the underlying infrastructure provider, to a specific use case rather than a customer.

For instance, we consider a quota of resources being assigned, let's say to an enhanced mobile broadband (eMBB) use case. Such quotas are being assigned across several use case areas that the infrastructure provider intends to offer to customer (others could be massive machine-type communication). Within said use case quota, customers are now assigned resources that they can use from the overall resource pool. This assignment is the result of a business relationship between the infrastructure provider and the specific customer. For instance, a quota of 60% infra resource (we focus here on the compute resources but can easily extend this to the communication resources as well) for eMBB is now assigned to customer X within the defined network slice for eMBB, while 40% are being assigned to another customer.

The name-based routing being realized in the service routing solutions of [11] and being used in our platform, now allows for 'pinning' service requests of specific customers, defined as *service contexts* and expressed through, e.g., subscriber IDs or mobile operator IDs. This pinning is achieved through flexible context-specific sub-naming of resources according to the assigned quota and chaining the customer-specific service requests along those defined contexts. The reactive (pub/sub-based) name registration and update mechanism outlined in [11] is key to realizing such pinning capability in a short timespan rather than relying on DNS-based solutions. Ultimately, the shortest path forwarding to the nearest (now context-dependent) service instance ensure the adherence to the assigned quota, establishing a 'sub-slice' within the overall 'super slice' established for the overall use case. Such lightweight mechanism for recursive slicing is contrasted against customer-specific slices, reducing scalability in terms of slicing overhead with expected setup times being in the range of less than several hundreds of milliseconds. We expect a more thorough analysis of this approach in the context of recently started work in the NGMN SBA working group, which has liaised with the 3GPP to put forward solutions in this space.

3) Trials

The validation and evaluation of the architecture and platform is conducted through over 20 trials that bring together 5G infrastructure operators, service providers and end-users to understand the full-stack assessment of performance, feasibility and acceptance. The trial process includes establishing 5G infrastructure including mobile edge and hierarchical data centres within in real-life urban settings and then provisioning a 5G infrastructure slice for platform deployment to create a testbed for trials and experimentation. Service providers then design micro services using service design patterns constructed to deliver enhanced user experience through platform features, and finally conduct technical trials and user evaluation on the testbeds.

Replication sites have been established in Bristol and Barcelona supporting real-life deployment in urban settings. Each site offers a radio access network along with hierarchically scaled computing clusters across a significant area of the cities. The virtual infrastructure manager is based on OpenStack and switches compliant with the OpenFlow 1.5 specification. Tenant slice specifications have been defined describing the infrastructure topology and capacity, that have then been used by an automated toolchain to provision platform services and establish the testbeds themselves.

A series of trials driven have been conducted by service providers in content rich industries seeking to improve user experience and reduce cost of service delivery through flexible placement and connection of services anywhere within the network from the far edge to the distant cloud. Trials have covered a broad range of scenarios such as content mobility for highly mobile users, optimized asset distribution for localized content in augmented reality applications and games, localized broadcast including participatory models for citizens and public broadcasters, and a variety of edge processing scenarios to understand and control content locally within highly distributed content production workflows that include many publishers.

In all trials, service providers have needed to shift paradigm from mobile cloud to mobile edge computing through new deployment and scaling options, including entirely localized services that do not rely on distant public clouds. Micro-service design patterns have emerged for mobile edge computing supporting enhanced user interaction such as opportunistic multicast, synchronized playout, nearest playout, proxy cache playout, content placement, application function offloading and geographic scaling. Using the design patterns the trials have demonstrated significant cost reductions and enhancements to user experience. Firstly, devops costs are reduced as services are now flexible, robust and agile through the platform dynamically placing, booting and connecting services in the network in response to demand and endpoint/routing policies. Secondly, network costs are reduced due to reduced network usage when multicast

delivery or content frontloading in segments of the network, whilst server costs are reduced through dynamic replication of content within the network and reduced server usage as http request suppression when multicast occurs. Finally, user experience is improved through reduction in start-up time for playout, battery life can be increased by offloading functions to the network, reaction time to latency changes is improved, latency stays comparable even when client moves between access points, whilst content is still reachable from anywhere in the network.

VI. Conclusion

In this paper, we presented our platform solution as well as first insights of deploying 5G services, both at the control as well as user plane, in cloud-native environments. For this, we outlined main concepts, components and lifecycle management cycles necessary to realize our ideas. The availability of the platform is essential for trial-based engagement pursued in a number of public funded projects but also driving the technological solutions into key standardization efforts. Through this, we complement our technology and platform development with evidence-based experimentation which in turn amplifies our efforts in standards and commercial exploitation with our platform already been recognized as one of three possible deployment options for upcoming 5G systems based on the most recent Release 16 specification.

In our future work, we plan on extending this standard basis through incorporating new ideas for resource management, realizing our 'enhanced slicing', presented in Section V.C.2) in control and user plane scenarios. Key to such flexible resource management will be the study of more complex policy decisions that will utilize the CLMC-based analytics built into our platform, which in turn will enable more complex edge computing scenarios with, e.g., migration of service function endpoints due to changing conditions at user and network level.

VII. Acknowledgements

This research is supported by the FLAME project, European Commission grant H2020-ICT-2014-1/731677. We acknowledge support of other FLAME consortium members.

References

[1] 3GPP TR 23.501, "System architecture for the 5G System (5GS), v16.0.0"; Rel. 16; Mar 2019.

[2] Fehling C, Leymann F, Retter R, Schupeck W, Arbitter P.; "Cloud computing patterns: fundamentals to design, build, and manage cloud applications"; Springer Science & Business Media; 2014 Feb 18.

[3] Leymann CF, Retter R, Schupeck W, Arbitter P.; "Cloud computing patterns"; Springer, Wien; doi. 2014;10(2014):978-3.

[4] Wilder B; "Cloud architecture patterns: using microsoft azure"; O'Reilly Media, Inc.; Sep 2012.

[5] 3GPP TR 23.742, "Study on Enhancements to the Service-Based Architecture, v16.0.0"; Rel. 16; Dec 2018.

[6] J. Halpern (Ed.), C. Pignataro (Ed.); "Service Function Chaining (SFC) Architecture"; RFC 7665; https://tools.ietf.org/html/rfc7665; IETF; Apr. 2015.

[7] ICT-FLAME Consortium; "D3.10: FLAME Platform Architecture and Infrastructure Specification V2", https://www.ict-flame.eu/deliverables/; Dec. 2018.

[8] OpenStack Foundation, OpenStack (Online, available: https://www.openstack.org).

[9] Open Networking Foundation, OpenFlow (Online, available: https://www.opennetworking.org).

[10] Organization for the Advancement of Structured Information Standards, "OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.1"; Jan 2018.

[11] "Service-based Routing at the Edge", Dirk Trossen, Sebastian Robitzsch, Scott Hergenhan; Available on arxiv at https://arxiv.org/abs/1907.01293, 2019.