Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH LETTER

5G Service Delivery Platform: 360 VR Vertical as an Example

Kay Hänsge[1] | Dirk Trossen[2] | Sebastian Robitzsch[2] | Michael J. Boniface[3] | Stephen C. Phillips[3]

[1]InterDigital Germany GmbH, Berlin, Germany
[2]InterDigital Europe Ltd., London, United Kingdom
[3]IT Innovation Centre, Southampton, United Kingdom

Correspondence
*Email: kay.hansge@interdigital.com

Summary

Operators have been adopting the cloud-native paradigm for the roll-out 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 increased flexibility, meaning new business cases may be addressed while maintaining competitive cost levels, and also the ability to realise use cases which historically have required the full breadth of infrastructure level knowledge. We present a realization of a service delivery platform entirely built upon service-based architecture principles.

KEYWORDS:
service-based architecture, service routing

1 | 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.The main motivation for this paradigm shift is that of increased flexibility in realizing new services (i.e. anything is a service) while achieving similar or even better cost economics as observed in the over-the-top cloud computing space. The use cases considered in standard efforts demonstrate this expected benefit, specifically the realization of novel control planes (through a set of micro-services), while also enabling an improved data plane through programmable services placed deep in the network, all of which communicate via a single communication platform, which in turn are provided via distributed cloud resources over 5G connectivity. A 'cloud-native' application is designed specifically for a cloud computing architecture. It leverages the benefits of cloud computing frameworks that compose loosely coupled cloud services towards such application. For this, 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 Next Generation Mobile Networks (NGMN) forum and the accompanying efforts in the 3rd Generation Partnership Project (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. However, most envisaged user plane services originate from public cloud offerings and are not familiar with 3GPP systems that require a vast amount of telecommunication knowledge[1]. The realization of such abstraction is captured as one of three possible deployment choices for 5G cloud-native service platforms in the relevant 3GPP standard for the 5G system (Appendix G4)[1].

The remainder of the paper is structured as follows: Section 2 introduces the platform, followed by the lifecycle management and control in Section 3. A trial description in Section 4 is presented before the paper is concluded in Section 5.

[1]https://www.3gpp.org/news-events/2110-changing-3gpp-to-help-the-verticals

2 | CLOUD-NATIVE DEPLOYMENTS

2.1 | Relationships and Interactions

Figure 1 outlines the relationships in the cloud-native deployment of services, following the well-established model found in data centre deployments[2,3]. The Figure also highlights the number of related entities. 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.

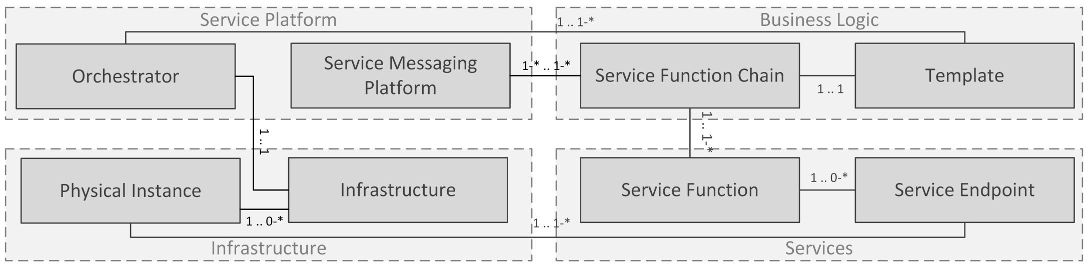

FIGURE 1 Service interaction pattern[4]

The glue between the virtualized services and the infrastructure is a Service Messaging Platform (SMP). It is this SMP that is the focus of ongoing 3GPP work, entitled as Service Communication Proxy (SCP) within the enhanced Service-based Architecture (eSBA) key activity within 3GPP[1].

The baseline is built upon multiple collaboration projects among European partners such as FP7 PURSUIT (Publish Subscribe Internet Technology)[2], H2020 POINT (iP Over IcN– the betTer IP)[3] as well as Horizon 2020 RIFE (architectuRe for an Internet For Everybody)[4]. The general focus on the service delivery level was to use Information-centric networking (ICN) techniques to forward request and responses of data. The next step was to introduce management and control functionality for Service Functions (SFs), always in alignment with the Service Interaction Patterns, presented in Figure 1.

Figure 2 depicts the overall service platform architecture[4], embedded into the relationships already shown in Figure 1. 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. This realization is based on 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). The platform itself is working in tenant mode and therefore respects other tenants that might using the same infrastructure. OpenStack is being used to deploy the presented components of the platform as well as cluster runtimes on different locations including those that can be geographically distributed; those cluster runtimes are then used by the platform to deploy Service Function Endpoints. Between the clusters and all deployed platform components, an SDN-enabled switching fabric performs path-based forwarding of packets.

The platform offers its own cloud-native orchestrator allowing to orchestrate services on top of the eSBA platform. Hence, service providers do not directly interact with the infrastructure. Instead, they express their deployable Service Function Chain (SFC) and requirements through templates through TOSCA-like descriptor, allowing for an automated process to deploy compute, storage and networking elements.

The Service Function Endpoint Management and Control (SFEMC) component processes the computational instance requests from the orchestrator and maintains lifecycle states of the deployed endpoints. It decides, based on the initial deployment policies defined in the orchestration templates, whether and on which infrastructure location (namely host or cluster) the described

[2]http://www.fp7-pursuit.eu
[3]https://www.point-h2020.eu
[4]https://www.rife-project.eu

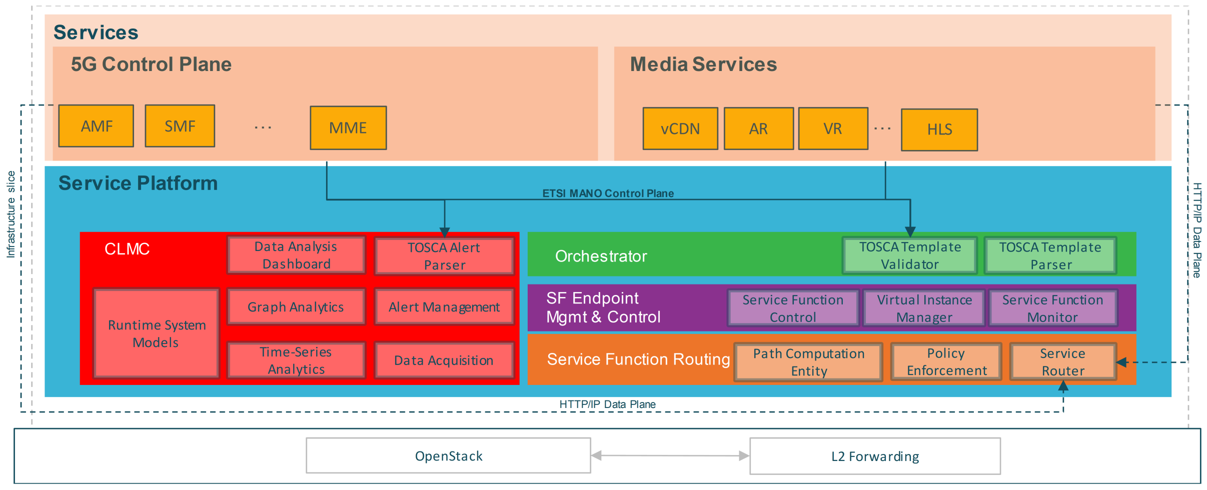

FIGURE 2 Platform architecture

instances may be deployed. When triggered from external sources (e.g., Cross-Layer Monitoring and Control (CLMC) or Service Providers), the SFEMC performs targeted transitions of the deployed lifecycle state of each Service Function Endpoint (SFE) and applies lifecycle policies defined in the templates against it.

The Service Function Routing (SFR) component 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 Fully Qualified Domain Name (FQDN) 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[5].

The CLMC component provides monitoring, analysis and control for adaptive 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.

2.2 | Business Logic

For the management as well as control of service deployments, the business logic is captured in the form of a cloud-native templates (descriptors). The instructions inside those descriptors are formulated along a SFC which in turn defines an ordered set of abstract SFs and ordering constraints that must be applied to the underlying routing[6]. SFs themselves are pre-packaged for a specific hypervisor (e.g. LXC or KVM) and once instantiated on a compute resource defined as a SFE (instance of SF). Service instances running on the platform require a lifecycle model which is managed by the platform orchestrator and fully decoupled from the infrastructure's orchestrator. For this to work the platform and infrastructure player maintain an agreed resource pool which underlies a well-defined business relation (expressed through a resource quota and SLAs). Beyond the initial SFC deployment the platform orchestrator also controls the runtime behaviour of the service endpoints. Said behaviour being defined as part of the initially provided template and changes triggered through cross-layer monitoring and analytics.

2.3 | Chain Specification

The SFC is described with a Topology and Orchestration Specification for Cloud Applications (TOSCA) for NFV template[7] that defines the SFs and their properties (nodes) and where and how SFEs should be deployed (policies). While TOSCA is widely used when describing services within ETSI/NFV terminology its ability to define relationships between nodes (SFs) does not allow the flexibility required to decouple the routing between SFs from their description. As a consequence, build plans and relationships are fully taken out of the revised descriptor[8].

3 | LIFECYCLE MANAGEMENT AND CONTROL

For each Service Function Endpoint, the platform maintains a runtime lifecycle in the form of a state machine, depicted in Figure 3, where Endpoints may be set in either *non-placed*, i.e. the SFE is only known to the network but there is no placement of the image on any of the available clusters; *placed*, i.e. the SFE is available on the targeted clusters and occupies disk space; *booted*, i.e. the SFE is running on the cluster and if needed, the SFE may prepare the actual service; *connected*, i.e. the SF Endpoint is discoverable and may receive service requests over the platform's network.

All states also have transition phases which can be used as performance indicators. Such transition states result in event driven notifications towards a CLMC, a data storage or other monitoring elements. As example the target state for an SFE is *booted*, hence the SFEMC will command to boot-up a new SFE at the destined location using a platform-dependent *boot* command. The CLMC observes such actions and events. When reaching *booted*, an event is triggered that this specific instance is now in the targeted state.

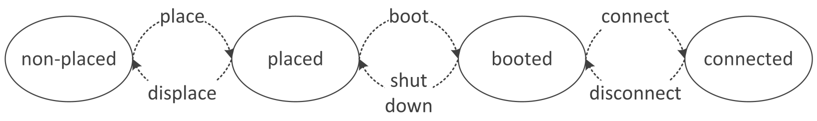

FIGURE 3 Lifecycle state machine of an SFE

Through the states described above, the resource consumption and lifecycle performance of each SFE can be controlled across the dimensions of storage, compute and communication resources. The SFE state is set up as a result of the orchestration actions, while being controlled over time via policies that are defined in the orchestration template.

3.1 | Management

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 virtual machine image including installation and setup of all required software and platform specific management services. Finally, the service function packages are published into a repository that is accessible to the platform and supports runtime discovery during service provisioning.

After the packaging is finished and uploaded to a image repository, 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 in 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 defined by the initial policy.

3.2 | Control

When an SF Endpoint enters the BOOTED 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 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.

Furthermore, *alert trigger* events are defined in the orchestration template for CLMC and SFEMC 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 information defines the conditions for triggering an alert that includes the metric and threshold for the critical value compared to the measured. 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 SFEs 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.

4 | 4K 360-VIDEO VIRTUAL REALITY TRIAL EXPERIENCE

Back in July 2019, a showcase day of the H2020 FLAME project[9] had been organised in Bristol, UK where the service delivery platform described in this paper enabled a 4k 360-video Virtual Reality (VR) trial. Figure 4 illustrates the deployed platform with the turquoise boxes representing Virtual Network Functions (VNFs) implementing the service routing and the orange boxes being standard IP endpoints. Note, the boxes called *cluster* allow the platform's cloud-native orchestrator to deploy SFs using one of the four states, as described in Section 3.2. Each ellipse box represents a hardware SDN switch and the ones labelled `t*` are located in one of the towers on Millennium Square in Bristol city centre (and so are the turquoise and oranges boxes connected to these switches). Each tower opens a dedicated WiFi access point beacon allowing clients on the square to select on of the eight locations.

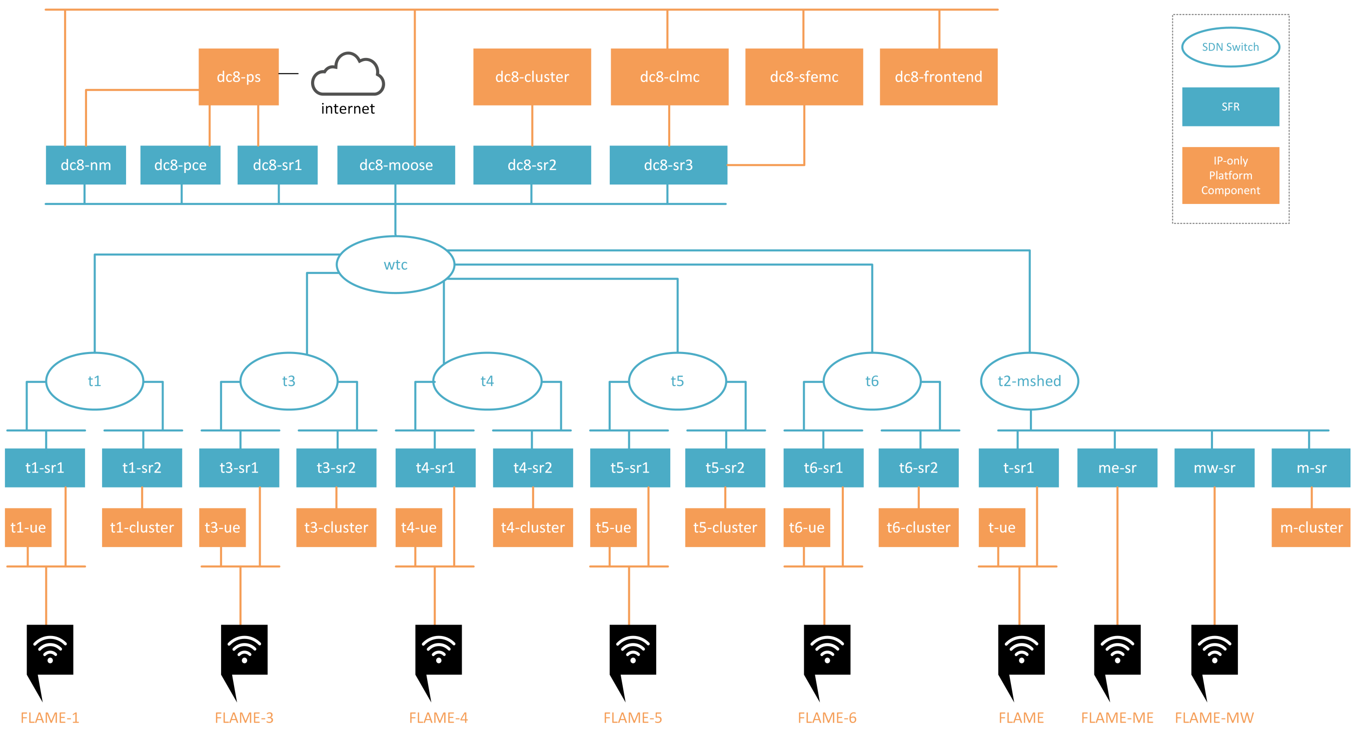

FIGURE 4 Deployed platform in Bristol, UK

The VR service deployed for the trial in Bristol implemented a tourist guide scenario where all visitors either have a VR headset or a planar view application and open the same video. The guide then controls start/pause/play of all clients and they are synchronised in segments over time utilising Dynamic Adaptive Streaming over HTTP (DASH). The service was composed of two SFs: the synchronisation SF and the storage SF hosting the VR content. While the sychronisation SF was deployed at the data centre only (dc8-cluster instance in Figure 4), the storage SF was deployed at each tower in state *booted* and at the data centre in `connected` state. As all storage SFEs are registered under the same FQDN against SFR, all clients receiving the start command from the tourist guide get the DASH segments provisioned by the nearest (in terms of number of hops) SFE. Once the CPU load of the storage SFE went above 70% for five seconds a scale out horizontally policy is being triggered to set all tower (edge) storage SFEs to *connected* allowing to serve clients at each tower from a dedicated SFE. In order to make this work for the service provider was to use the cloud-native orchestration template to define the properties of each SF (including their FQDN) and to define the state of each SFE.

5 | 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 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.

ACKNOWLEDGEMENT

The work presented in this paper has been performed in the framework of the FLAME project. This project has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 731677.

References

1. 3GPP . System architecture for the 5G System (5GS), v16.0.0. Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP); 2019. Version 16.0.0.
2. Fehling C, Leymann F, Retter R, Schupeck W, Arbitter P. *Cloud Computing Patterns*. Springer Wien . 2014.
3. Wilder B. *Cloud Architecture Patterns*. O'Reilly . 2012.
4. Trossen D, Robitzsch S, Haensge K, Boniface M. D3.10: FLAME Platform Architecture and Infrastructure Specification V2. tech. rep., ICT-FLAME Consortium; 2018.
5. Trossen D, Robitzsch S, Hergenhan S. Service-based Routing at the Edge. https://arxiv.org/abs/1907.01293; 2019.
6. Halpern JM, Pignataro C. Service Function Chaining (SFC) Architecture. RFC 7665; 2015
7. TOSCA Working Group . OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.1. tech. rep., Organization for the Advancement of Structured Information Standards (OASIS); 2018.
8. Martín CA, Hernàndez J, Hänsge K, et al. D4.2: Report on Components for Adaptive Service Delivery. tech. rep., ICT-FLAME Consortium; 2019.
9. ICT-FLAME Consortium . Citizen Services, Gaming and Immersive Media Showcased on a New 5G Platform in Real-life Trials. https://www.ict-flame.eu/news/citizen-services-gaming-and-immersive-media-showcased-on-a-new-5g-platform-in-real-life-trials/; 2019.