Separation of concerns among application and network services orchestration in a 5G ecosystem

Panagiotis Gouvas, Anastasios Zafeiropoulos, Eleni Fotopoulou, Thanos Xirofotos
Network Softwarization and IoT Group,
UBITECH,
Athens, Greece

Roberto Bruschi[*], Franco Davoli[*,°]
[*]CNIT S3ITI National Laboratory
Genoa, Italy
[°]DITEN – University of Genoa
Genoa, Italy

Anderson Bravalheri, Dimitra Simeonidou
High Performance Networks Group,
Department of Electrical and Electronic Engineering,
University of Bristol,
Bristol, UK

***Abstract*—5G networks evolution is tightly bounded with the need to support vertical industries network performance and operating requirements. Such an objective is associated with a set of challenges related mainly to the provision of frameworks that can tackle aspects related to the various 5G stakeholders (e.g. software developers, application/service providers, telecom/infrastructure providers), without imposing strict requirements on their collaboration terms. Towards this direction, in the current manuscript, we detail a novel holistic framework tackling the overall lifecycle of 5G-ready applications design, development, deployment and orchestration over application-aware network slices. Separation of concerns among the related stakeholders per layer of the proposed architecture regards one of the basic considered principles.**

Keywords—Service Mesh; Cloud Native; NFV orchestration; Mobile Edge Computing; Intelligent Orchestration; Network Slice

I. INTRODUCTION

One of the main objectives of the 5G technology specification certainly resides in the enablement and support of a new class of vertical applications with very heterogeneous but extremely challenging performance and operating requirements. However, to achieve the integration of verticals over a highly evolving and heterogeneous networking and computing ecosystem, vertical industries' needs should be considered as drivers of 5G networks design and development with high priority.

In the current manuscript, a novel and holistic approach is presented, tackling the overall lifecycle of applications' design, development, deployment and orchestration in a 5G ecosystem, as it is under development within the framework of the MATILDA H2020 project [1]. A set of novel concepts are introduced, including the design and development of 5G-ready applications based on the introduction of a set of principles adopted towards the design of cloud-native applications, the separation of concerns among the orchestration of the developed applications and the required network services that support them, as well as the specification and management of network slices that are application-aware and can lead to optimal application execution.

As 5G-ready application, we consider a distributed application consisting of independently orchestratable cloud-native components able to take advantage of both the programmable network and computational infrastructure. Actually, a 5G-ready application regards an application developed based on cloud-native development patterns, while declaring set of requirements to be fulfilled by the underlying infrastructure. A 5G-ready application is represented in the form of an application graph that includes the set of application components along with their interconnection. Interconnection is realized in terms of binding among software-denoted required and exposed interfaces. Based on the design and development of a 5G-ready application metamodel [2], a set of deployment and operational requirements are declared by software developers. Such requirements lead to the specification of deployment constraints that must be fulfilled during the deployment of the application over the programmable infrastructure. They also facilitate application/services providers to specify effective runtime policies over the developed software (e.g. load balancing policies over software components that are declared as horizontally scalable). Given the specification of the set of deployment and operational requirements on behalf of the application developer within a descriptor, no further actions are required on its part.

Separation of concerns is realized towards the orchestration of the provided vertical applications as well as the required network services. In the provided approach, orchestration of the application graph is realized by a vertical applications orchestrator, following a service-mesh-oriented approach, while orchestration of network services is realized by network management mechanisms over an instantiated network slice.

The recently introduced service mesh concept [3] is adopted for vertical applications orchestration, as a software management layer for controlling and monitoring internal, service-to-service traffic in microservices-based applications. It

consists of a data and a control plane. The data plane consists of a set of intelligent proxies deployed alongside the application software components supporting the provision of support/backing services (e.g. service discovery, load balancing, health checking). The control plane manages the set of intelligent proxies based on distributed management techniques and provides policy and configuration guidance for all the running support/backing services [3].

The concept of network slice [4] is used for deployment and management of the required network services based on vertical application needs. A network slice is a logical infrastructure partitioning allocated resources and optimized topology with appropriate isolation, to serve a particular purpose of an application graph. During a vertical application's deployment request, based on the denoted requirements by the vertical application orchestrator, instantiation and management of the appropriate application-aware network slice takes place. These actions belong to the communication service provider domain and can be realized in an agnostic way to application service providers.

In the current manuscript, the overall approach is presented, focusing mainly on the principles that must be adopted by application developers towards the design of 5G-ready applications. The structure of the manuscript is as follows: in section two, the main artefacts that have to be considered towards the design of cloud-native applications are detailed; following, in section three, the overall architectural approach for supporting the placement of such applications over 5G network slices promoting the separation of concerns among software developers, application/service providers and communication service providers is described; finally, section four provides main conclusions and challenges to be tackled on future work.

II. Cloud-Native Applications and Network Services Development

As already mentioned, the basic set of principles considered towards the design and development of 5G-ready applications stems from the specification of cloud native applications. Cloud native applications are designed and developed in an infrastructural-agnostic manner and mainly built based on a cloud computing model. They regard a combination of existing (software automation, API integrations, services-oriented architectures) and new software development patterns (microservices architecture, containerized services, distributed management and orchestration) [5]. They can be deployed and managed providing organizations with greater agility, resilience, and portability across cloud environments. However, in order to be deployed and executed in an optimal way, there is a need for support of automated and dynamic orchestration aspects as well as dynamic configuration of the programmable infrastructure, along with continuous monitoring and management of the allocated programmable resources. In a 5G ecosystem, software development based on cloud-native principles may regard both the development of vertical applications as well as the development of virtual network functions (VNFs), given the need to tackle similar challenges by orchestration mechanisms in different layers.

Within MATILDA, a 5G-ready application metamodel is specified, considering the twelve-factor app methodology [6] that is considered the main methodology leading to the design and development of cloud-native applications. This metamodel targets the development of vertical applications; however, most of the concepts are also applicable for VNFs' development. In addition to the twelve-factor app methodology, properties highly relevant to intelligent orchestration mechanisms and the programmability of compute, storage and network resources- are introduced. A set of important aspects about orchestration and programmable infrastructure management are also denoted.

Regarding orchestration, the first critical issue is chainability. Chainability is achieved by enabling interconnection among software components and is declared in the form of dependencies. Such dependencies are denoted in terms of required and exposed chainable interfaces. Each software component has to provide a set of exposed and required interfaces that can be used for binding with other software components, leading to the creation of application graphs. A dynamic coupling of interfaces between two software components is highly valuable only when an actual binding can be fully automated during runtime. This level of automation enforce strong constraints for the developed software components. The 'profile' of the chaining should be clearly abstracted. Such profile includes the offered/required datatype, the ability to accept more than one chain etc. These metadata are often stored in high-efficient key-value stores in order to be queried by requesting software components. It should be noted that part of the chainable interfaces may provide access to service mesh or support/backing services, as specified in the twelve-factor app. A support/backing service is a service consumed by the application over the network as part of its normal operation. Based on common definition of interfaces, no distinction between local and third-party services applies.

Another critical issue is configurability. More specifically, each part of the application has to be (re)configurable-by-design (i.e., to adapt to the new configuration without interrupting its main thread of execution). Configurability takes place not only during the deployment of the application, but also during runtime through the specification and exposure of mutable configuration parameters that can be modified during runtime through a configuration server. In both cases, configuration options have to be made available in a descriptor, based on a set of environment variables, that are easy to change between deployments without changing any code, and also language- and OS-agnostic.

Software components should be executed as one or more stateless processes in order to be horizontally scalable by design. Any data that needs to persist must be stored in a stateful backing service, typically a database or a high-performance cache. Any service that is stateless can scale easily with the usage of some support/backing services such as network balancers or web balancers. Historically, these services were statically configured by administrators or by DevOps engineers. The emergence of the infrastructure programmability model will progressively 'offload' this task to service mesh functions that are controlled by an orchestrator.

Ensuring the stateless behaviour of an application graph is a challenging task since the entire business logic should entail stateless behaviour.

Each software component has to be able to be managed and scale independently. Software component lifecycle management (e.g. start/stop, restart upon failure, fast start-up), as well as scaling and deprovision management (e.g. add new instances, graceful shutdown), have to be supported. Lifecycle management actions may be applied at any moment, facilitating elastic scaling, rapid deployment of code or config changes, and robustness of production deploys. If possible, processes should also be robust against unexpected runtime execution failures, in the case of a failure in the underlying infrastructure.

To enable traceability and logging, each software component's streams have to be captured by the execution environment, collated together with all other streams from the application, and routed to one or more final destinations for viewing and long-term archival. These archival destinations are not visible to or configurable by the application, and instead are completely managed by the execution environment. Collection of such monitoring data can be meaningful for introspecting application's behaviour over time and thus for profiling purposes. Traceability in terms of identification of faults and misbehaviours of software components has also to be supported.

With regards to the programmable infrastructure management aspects, there are many issues that can be modelled. More specifically, each software component has to expose quantitative metrics regarding the required Quality of Service (QoS) level to be provided that may regard resource-usage metrics (e.g. VCPU, memory, network usage) or component-specific metrics (related to the business logic of the application, e.g. number of sessions, average response time). The first set of metrics are monitored by appropriate infrastructure monitoring probes by exploiting the infrastructure programmability aspects, while the latter through the implementation and exposure of application-level probes (e.g. supported in the service mesh telemetry functions).

Furthermore, a component should be developed in a way that it does not impose strict limitations with regards to the infrastructure that is going to host it, thus it has to be programmable and infrastructure agnostic. The provided software has to be portable to any modern programmable infrastructure, avoiding software development patterns that impose such limitations. Packaging of a software component in multiple formats have also to be supported (e.g. as a container, virtual machine or unikernel where applicable). Finally, the developed software components should be able to take advantage of the programmability layer provided by infrastructure providers (including communication service providers and cloud service providers). Functions related to live migration, placement considering locality aspects, mobility, VPN establishment, vertical scaling, etc., can be exploited, given the description of such needs in the application part. For instance, a software component that is running on a specific infrastructure may be literally 'transported' to another one without any down-time (in case of a live migration), without significantly affecting the overall performance of the provided application (consisted of a set of software components).

III. MATILDA Overall Architectural Approach

A. Reference Architecture

MATILDA reference architecture is divided in three distinct layers (as depicted in Fig. 1); namely the Applications Layer, the 5G-ready Application Orchestration Layer and the Programmable 5G Infrastructure Slicing and Management Layer. Separation of concerns per layer is a basic principle adhered towards the design of the overall architecture. The Applications Layer is oriented to software developers, the 5G-ready Application Orchestration Layer is oriented to application/service providers and the 5G Infrastructure Slicing and Management Layer is oriented to communication service/infrastructure providers.

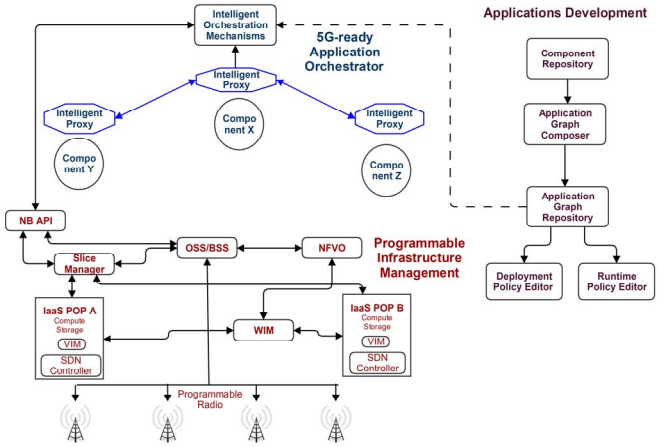

Fig. 1. MATILDA Reference Architecture.

The Applications Layer takes into account the design and development of 5G-ready applications per industry vertical, along with the specification of the associated networking requirements. The associated networking requirements per vertical industry are tightly bound together with their respective 5G-ready applications' graph, which defines the business functions, as well as the service qualities of the individual application.

The 5G-ready Application Orchestration Layer supports the dynamic on-the-fly deployment and adaptation of the 5G-ready applications to its service requirements, by using a set of optimisation schemes and intelligent algorithms to provide the needed resources across the available multi-site programmable infrastructure.

The Programmable 5G Infrastructure Slicing and Management Layer is responsible for setting up and managing the 5G-ready application deployment and operation over an application-aware network slice. Network slice instantiation and management, network services and mechanisms activation and orchestration as well as monitoring streams management are realized. Such actions are triggered based on requests

provided by the 5G-ready Application Orchestration Layer through the specification of Open APIs.

B. *Applications Layer*

Software development encompasses application components, overall application graphs, virtual network functions or network services. In each case, adherence to a relevant metamodel defined within MATILDA has to be fulfilled. For vertical applications development, the relevant metamodel is the 5G-ready application graph metamodel, while for VNFs and NSs development, an ETSI NFV [7] specifications compliant metamodel has been adopted. In both cases, suggestions, guidelines as well as metamodel artefacts for developing software based on cloud-native principles are provided. The objective is to foster software that can be independently and easily maintainable and extensible, able to take advantage of the supported intelligent orchestration and programmable infrastructure management mechanisms. Validation mechanisms take place for ensuring the appropriate software development concerning the defined metamodels and cloud-native principles. The developed software components (application components, VNFs) are made available in a software component repository. Finally, through a graph composer, application graphs and network service graphs can be created and made available to service providers.

C. *5G-ready Application Orchestration Layer*

The 5G-ready Application Orchestration Layer is responsible for deploying and managing the overall orchestration of 5G-ready applications over network slices realized by the Programmable 5G Infrastructure Slicing and Management Layer (see Fig. 2). A set of intelligent orchestration mechanisms are specified tackling optimal deployment, runtime policies enforcement, provision of service mesh control and data plane functionalities, real time monitoring and big data analysis aspects.

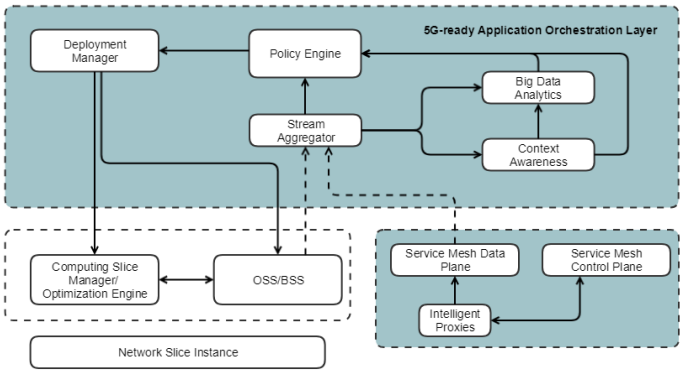

Fig. 2. 5G-ready Application Orchestration Layer.

Upon the receipt of a vertical application deployment request, the Deployment Manager is responsible to process the software descriptors along with any further deployment constraints and objectives denoted on behalf of the services provider and proceed to the request for the instantiation of the appropriate network slice (see Fig. 3). This request is tackled by the Slice Manager that resides in the Programmable 5G Infrastructure Slicing and Management Layer and does not require any interaction with the 5G-ready application orchestration mechanisms. Upon the instantiation of the network slice -that includes the activation of the required network mechanisms and the reservation of the required computational and storage resources-, the deployment process takes place, leading to the placement of the application graph and the provision of the application functionalities. Real time monitoring streams are activated towards a Stream Aggregator. Such streams are coming from monitoring mechanisms by the application's intelligent orchestration mechanisms (including service mesh telemetry mechanisms) or by end-to-end network monitoring mechanisms supported by the communication service provider. Activation of the latter streams can be realized through open northbound APIs provided on behalf of the communication service provider.

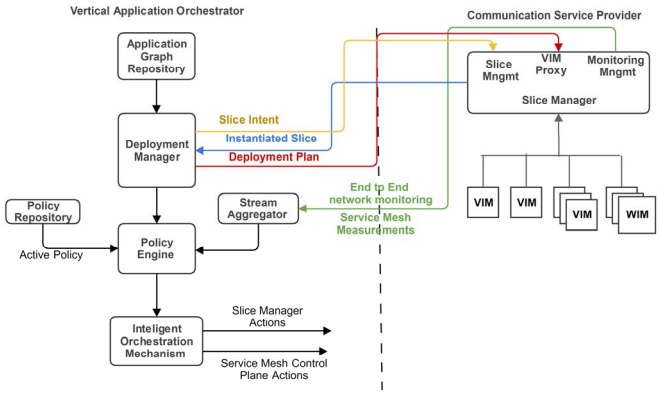

Fig. 3. Separation of concerns among orchestration domains.

Following, runtime policies enforcement takes place leading to runtime activation and management of set of functionalities. Within MATILDA, a service mesh approach is adopted for runtime management of the application graph and runtime activation and provision of set of support/backend services (e.g. load balancer, authorization, telemetry, layer 7 traffic management). The service mesh is separated in the control and data plane. The data plane is responsible for implementing the support/backend services through Intelligent Proxies adapted to each software component. The control plane is responsible for managing in a distributed way the provision of data plane services, taking into account the actions requested by the applied policies.

Thus, a wide set of the envisaged actions of the runtime policies enforcement mechanisms are going to regard service mesh-oriented actions. However, further programmable infrastructure-oriented actions are going to be supported, leading to real-time activation and management of network mechanisms through the consumption of relevant open APIs. It should be noted that the set of orchestratable cloud-native characteristics supported by the developed vertical applications, are exploited in this layer.

D. *Programmable 5G Infrastructure Slicing and Management Layer*

The Programmable 5G Infrastructure Slicing and Management Layer (see Fig. 4) supports a set of network

management functionalities, based on the deployment and runtime requests provided by the 5G-ready Application Orchestration Layer. It consists of the Slice Manager, the Business and Operational Support Systems (BSS/OSS), the NFV Orchestrator and the Mobile Edge Orchestrator (MEO), while programmable resources management is realized over a set of Virtualization Infrastructure Managers (VIM) and Wide-area Infrastructure Managers (WIM). Interaction with the 5G-ready Application Orchestration Layer is realized through the provision of set of northbound open APIs.

The Business and Operational Support Systems (BSS/OSS) is providing resources of the communication service providers as-a-Service to vertical industries. According to the latest specifications in 3GPP [8], these modules will expose the 5G network to verticals in terms of 5G network slices. At the southbound, as specified in [7], the BSS/OSS are supposed to interface to the NFV Orchestrator (NFVO) to request the activation/deactivation/modification of NFV services, and to the VNF instances for configuration purposes. The NFV Orchestrator(s) (NFVO) is managing the network services composing the network slices activated by the BSS/OSS. In case of identified edge computing requirements, the Mobile Edge Orchestrator (MEO) is managing the embedding of mobile edge applications, and the management of their lifecycle. As specified in [9], the MEO is triggered by the BSS/OSS.

The Virtualization Infrastructure Manager(s) (VIM) are exposing the resources (especially computing and storage) of PoP datacentres mainly to the NFVO and to the MEO, as well as towards the vertical applications orchestrator. The Wide-area Infrastructure Manager(s) (WIM) are realizing the logical interconnectivity among sets of service/application components instantiated in different PoPs and/or towards 5G User Equipment (UE).

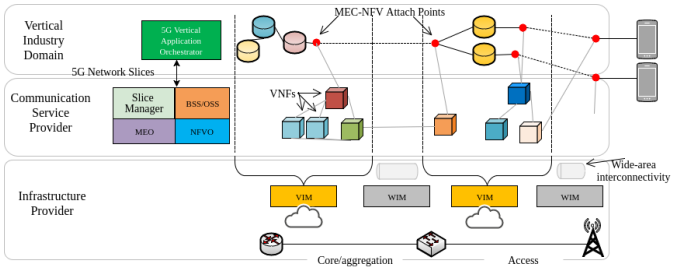

Fig. 4. Main elements of the Programmable 5G Infrastructure Slicing and Management Layer.

IV. Conclusions and Future Work

In the current manuscript, a novel holistic approach for the design, development, deployment and orchestration of 5G-ready applications over application-aware network slices is presented. The design and development of 5G-ready applications is based on the adoption of set of principles related to the development of cloud-native applications, as well as concepts related to independent orchestration and infrastructure-agnostic-oriented aspects. The detailed approach is based on three layers with distinct functionalities per layer and clear separation of concerns with regards to the involved stakeholders per layer, as well as any cross-layer interaction. The overall design is based on the specification of open interfaces and APIs as well as on the integration and exploitation of evolving open-source orchestration frameworks for applications (e.g. service mesh approaches) and network services (e.g. NFV orchestrators).

In our future work, there are plans for the detailed specification and instantiation of the suggested approach, as well as its validation and evaluation over a set of vertical industries domains (e.g. emergency communications, smart cities, media on demand, industry 4.0, automotive). Focus is going to be given on the design and implementation of novel mechanisms for translating application denoted network requirements to a slice intent that can lead to the instantiation of an application-aware network slice, the design and development of efficient and scalable intelligent orchestration mechanisms and the specification and implementation of set of open APIs for programmable infrastructure management.

Acknowledgment

This work has been co-funded by the European Commission MATILDA H2020 Project under the Contract H2020-761898.

References

[1] MATILDA H2020 Project, Available Online: http://www.matilda-5g.eu/

[2] MATILDA H2020 Project Deliverable D1.2 - Chainable Application Component & 5G-ready Application Graph Metamodel, Available Online: http://www.matilda-5g.eu/index.php/outcomes

[3] Pattern: Service Mesh, Available Online: http://philcalcado.com/2017/08/03/pattern_service_mesh.html

[4] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, "Network Slicing & Softwarization: A Survey on Principles, Enabling Technologies & Solutions," in IEEE Communications Surveys & Tutorials. doi: 10.1109/COMST.2018.2815638

[5] Developing Cloud Native Applications, Cisco Blogs, Available Online: https://blogs.cisco.com/cloud/developing-cloud-native-applications

[6] The twelve-factor app methodology, Available Online: https://12factor.net/

[7] ETSI GS NFV 002, "Network Functions Virtualisation (NFV); Architectural Framework," version 1.2.1, Dec. 2014. Available Online: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[8] 3GPP, "Study on management and orchestration of network slicing for next generation network," TR 28.801, version 15.0.0, Sept. 2017.

[9] ETSI GS MEC 003, "Mobile Edge Computing (MEC); Framework and Reference Architecture," version 1.1.1, March 2016. Available Online: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf.