

5G-Kube: Complex Telco Core Infrastructure Deployment Made Low-Cost

Domenico Scotece, Asma Noor, Luca Foschini and Antonio Corradi

***Abstract*—Network Function Virtualization (NFV) along with Software Defined Networking (SDN) have brought an evolution in telecommunications laying out the bases for 5G networks and its softwarization. Accordingly, new implementations of telecom standards, such as the 3GPP 5G Core, are defined as fully-virtualized infrastructures consisting of different components and leveraging a cloud-native approach. At the same time, standard-oriented solutions, such as ETSI Management and Orchestration (MANO), have emerged to master the complexity of Virtualized Network Functions (VNFs) orchestration, including 5G Core VNFs. While MANO operates at the NFV level, it also leverages existing cloud infrastructures for the deployment of VNFs by interoperating with resource orchestrators at the cloud level. From the business perspective, that requires telco operators to interact with different technology providers, from NFV/MANO software producers to cloud computing providers, and to hire technicians proficient in the technologies of both telco and computing worlds, that are a rather difficult human resourcing to find. The main claim of the paper is that the Development and Operations (DevOps) tools in the IT world are mature enough to leverage them directly in the telco world, without superimposing other interlaced standard/software. That allows to significantly reduce OPEX cost of complex telco infrastructures by supporting all needed automation and by avoiding the combined use of (too) complex layered standards/software stacks, such as in the case of MANO. Accordingly, in this paper, we leverage container-based technologies and Kubernetes to design and evaluate a novel deployment approach, called 5G-Kube, for softwarized 5G core networks. 5G-Kube, which is openly to the community, has been also evaluated in two different use cases of the 5G Core and Kubernetes deployment fitting, namely, Industry 4.0 and Smart Cities.**

***Index Terms*—SDN, NFV, Cloud Native, Kubernetes, 5G Core, 5G, Beyond 5G**

I. Introduction

Nowadays, the advent of Industry 4.0 along with the massive use of the Industrial Internet of Things (IIoT) have brought the production of a huge quantity of information. The demand for services and processing capabilities caused the need to bring innovation into the network infrastructure putting the basis the next-generation networks such as 5G networks and enabler paradigms like Software Defined Networking (SDN) and Network Function Virtualisation (NFV). 5G is being seen as an opportunity for transformative change with the transition to cloud infrastructure and a service-based architecture [1]. In particular, SDN allows the decoupling of the control plane from the data plane enabling the decentralization of the controller handling it as a collection of microservices hosted according to a cloud-native approach [2]. On the other hand, NFV brings softwarization capabilities to allow the implementation of hardware network functionalities as software packages running on a virtualized infrastructure. There has been an ongoing effort to define some standard technologies to implement these concepts but all of them present some lacks by leaving space to search for alternative approaches.

Regarding the 5G core, nowadays there are several implementation solutions both open-source and commercial that are not easy to deploy [3]. In particular, most of them have a disturbingly steep learning curve and require specialized staff. This is due to the fact that up to now the telco network infrastructure was often associated with telco provider solutions. With the advent of the 5G standard, the core should be deployed as software blocks on a cloud-native platform [4]. Furthermore, all deployment processes of the 5G core modules should be made automatic with the use of an orchestrator. The European Telecommunications Standards Institute (ETSI) has defined the standard solution for the management and orchestration named MANO. Moreover, ETSI provides an open-source implementation of MANO called Open Source MANO (OSM) [5]. Aligned with this direction, several solutions provided so

D. Scotece, A. Noor, L. Foschini and A. Corradi are with Department of Information Engineering and Computer Science (DISI) at University of Bologna, Bologna, Italy (e-mail: {domenico.scotece,luca.foschini,antonio.corradi}@unibo.it, asma.noor@studio.unibo.it).

far on the deployment of 5G Core started using OSM for the purpose of management, orchestration, and hosting [6]. However, the totality of existing solutions has a difficult entry level due to their complex structure requiring big efforts.

In this article, we present a *low-cost cloud-native 5G core deployment* solution using *Kubernetes as the manager and orchestrator of the 5G modules*. Moreover, our deployment solution is based on *microservices*, in particular, a dockerized version of the 5G core, and leverages most of the useful DevOps tools. With the advent of the GitOps concept [7], the 5G core deployment workflow is largely self-updating for the entire lifecycle of each application. Moreover, the presented solution, called 5G-Kube, facilitates the deployment of complex 5G Core deployments by using Kubernetes and makes the Kubernetes control plane fully aware of 5G core services. Additionally, we studied two different deployment scenarios for two use cases such as *Industry 4.0* and *Smart City*. Finally, we provide ready-to-deploy configurations for all network functionalities, already available for the community [1], over Docker Container technology and Kubernetes.

The rest of the article is organized as follows. In the next section, we give more details on the motivations and the state of the art of the 5G core implementations. Then we present our low-cost Kubernetes-based deployment solution and all possible scenarios. We present the results of all tests performed on the cases of deployments. Finally, we summarize the work done in this article and provide some future work directions.

II. Motivation and Background

The system architecture for the 5G system has been defined in 3GPP standard Release 15 with the definition of the Service Based Architecture (SBA) [8]. In particular, the 5G core modules are designed to be cloud-native and leverage advanced networking technologies such as NFV and SDN.

Naturally, OSM is the key enabler for deploying 5G Network Services (NSs). Unlike Physical Network Functions (PNFs) traditionally deployed on proprietary hardware, OSM supports the deployment of Virtual Network Functions (VNFs) on commodity hardware. Note that each NS is composed of one or multiple VNFs and these VNFs are connected by Virtual Links (VLs) and their interaction are presented as VNF Forwarding Graphs (VNFFGs) [9]. In general, MANO-based orchestrators require laborious work to manage the life cycle of VNFs, starting from the creation of virtual machine descriptors to the manual configuration of the software that will execute the network function. All platforms require a deep understanding of the underlying infrastructure and are complex to use [10]. In addition, interoperability could be an issue in those platforms, which in general are difficult to be integrated with other systems.

In our solution, we claim the use of Kubernetes as not only the Network Function Virtualisation Infrastructure (NFVI) but also the Network Function Virtualisation Orchestrator (NFVO), VNFs Manager, and Virtualized Infrastructure Manager (VIM). In particular, our solution does not use the OSM framework for orchestrating VNFs in order to considerably reduce the complexity. Furthermore, the 5G NSs will be implemented as *microservices* inside Docker containers which makes them more user-friendly. Solutions like [11], start to deploy the 5G core with Kubernetes in order to guarantee a high scalability level.

However, PNFs and VNFs have started the evolution process toward the concept of Cloud-Native Network Functions (CNFs). This is reflected in the transformation of network monoliths into microservices managed by Kubernetes. Generally, the MANO standard is supposed to be the orchestrator for the PNFs and VNFs by relying on virtual machines. Moreover, MANO, in contrast with the Kubernetes orchestrator, guarantees greater protection in terms of security aspects [12]. Nevertheless, thanks to the increased adoption of containerized CNFs by operators, Kubernetes is likely to become the standard orchestrator for the 5G core in the near future [12].

A. 5G Core Implementations

This section presents a selection of two of the most valuable open 5G core implementations available in the literature that we compare in Table I.

1) Open5GS: offers an open-source implementation of the 5G network core complaint to the 3GPP release 16 [13]. Some projects that are not official have started the cloud-native deployment of the Open5GS core implementation. The *open5gs-k8s*

[1] https://gitlab.com/dscotece/5gcore-kubernetes

TABLE I
OPEN SOURCE 5G CORE IMPLEMENTATIONS

Implementation	5G Core	EPC	License	Language	Docker support	Kubernetes support
Open5GS [13]	Release-16 compliant	YES	GNU AGPL v3.0	C-language	NO	NO
free5GC [14]	Release-15 compliant	NO	Apache 2.0	Golang	YES	NO

project from the community is still in progress but allows to partially deploy the core into a Minikube cluster. In parallel, a project named OpenVerso started to implement the Open5GS core with Helm a package manager for Kubernetes.

2) free5GC: offers an open-source implementation, written in Go language, of the 5G network core compatible with general-purpose machines that runs Ubuntu as the operating system [14]. Its implementation includes all the 5G modules of the core and, to the best of our knowledge, is one of the most completed already available. Furthermore, an early version of docker-based 5G core implementation is provided, named *free5GC-compose*, and it is based on some open-source implementations from the community. In particular, *free5gc-compose* provides an all-in-one implementation of the 5G core modules in docker-compose technology. Surely, it provides also the Docker settings for each 5G functionality.

As shown in Table I both solutions represent good candidates to be the starting point for implementing 5G-Kube. However, at the current stage, *free5GC* offers a slightly better base for Docker-based 5G Core deployment and that is the reason why we opted for it in our implementation. In addition to that, in this work we provide to the community 5G-Kube as a ready-to-deploy solution that eases both deployment and monitoring of 5G core modules making Kubernetes cluster aware of them.

III. KUBERNETES-BASED 5G CORE DEPLOYMENT

This section presents our 5G-Kube solution, by giving more high-level details on the architectural solution. Due to the strict space limitations in this paper, we canot present all the implementation details. On the one hand, this section provides an overview of the proposed architecture for low-cost 5G core deployment by using Kubernetes instead of the MANO orchestrator. On the other hand, we detail the cluster deployment for all the dockerized 5G modules.

A. Kubernetes Architecture

In recent years, Kubernetes emerged as the most successful container orchestrator. Its design lets users describe the desired data plane state of their applications and a control plane that works like a feedback control loop to drive the actual state toward the desired state, by achieving fault tolerance, self-healing, and large scale.

For those reasons, the Kubernetes controller can effectively handle the 5G core control plane capabilities by extending the Kubernetes control plane. In particular, this work put its emphasis on the data plane facilities by leveraging the scaling and scheduling features of the Kubernetes engine. 5G-Kube deployment provides provides an evolved Kubernetes control plane specifically tailored for 5G Core deployments as shown in Fig 1. In particular, we extended the controller plane of Kubernetes to manage/monitor 5G Core elements without requiring any change in the 5G Core network. At each worker node, we deploy our novel *5G-Kube agent* to support specific controller functionalities for the 5G Core network such as. The 5G API servers are the only components to interact with the 5G etcd cluster. Users manage the 5G control plane by creating the appropriate API objects in the 5G API servers, just like standard users of Kubernetes create, update, and delete their containerized applications. Moreover, Kubernetes allows service providers get a simplified architecture common for central, edge, and private network deployments which could lead to reduced capital expenditure (CapEx) and operational expenditure (OpEx) [15]. In particular, the use of Kubernetes allows managing not only the 5G core modules but also third-party services.

Focusing on the 5G-Kube data plane, there are severe benefits for operating Kubernetes for the 5G core deployment. First of all, it allows the use of containerized CNFs instead of VNFs, or better it allows running both VNFs and CNFs in the same infrastructure. This eliminates the need to port all of the applications to containers or to manage the two

implementations in separate stacks. So, in our test cases, we limited to the use of CNFs for the 5G core modules implementation. Second, Kubernetes automates all of the bare metal life-cycle management tasks. This enables 5G operators and developers to release the full performance of the physical hosts as an elastic and flexible bare metal cloud, where they can rapidly deploy and redeploy CNFs or VNFs. Third, Kubernetes provides advanced resource bindings required for 5G implementations as part of their SaaS delivery model. Telco providers and operators can offload the management of the CNFs life cycle (i.e., deployment, support, upgrades, monitoring) to Kubernetes. Finally, the deployment of 5G Core can be done in several different configurations - a first one in its full format containing all the modules defined in the standard and the others that are more essential and consist of only the main elements used for communication depending on the scenarios.

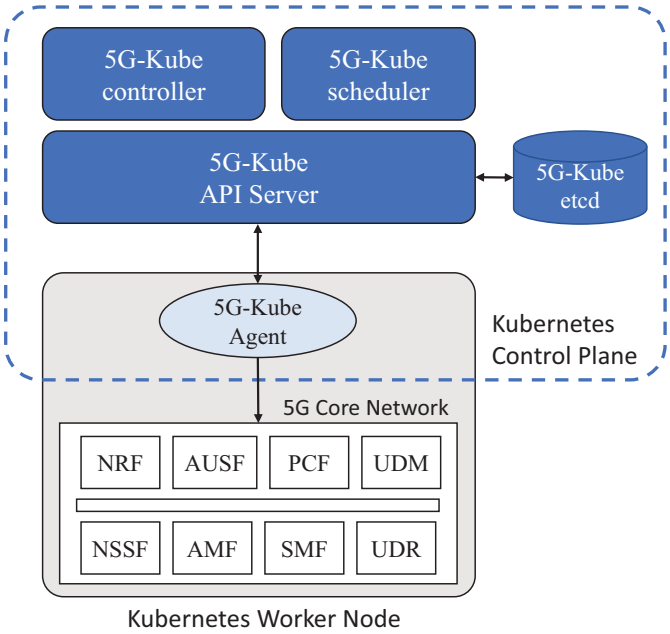

Fig. 1. Logical Architecture of the 5G-Kube Control Plane and Data Plane

B. Cluster Deployment

In this section, we illustrate how to implement the Kubernetes cluster in order to deploy a straightforward 5G Core implementation. Our early work on Kubernetes-based 5G Core relies on free5GC which provides a complete Release 15 3GPP-compliant 5G Core implementation. Additionally, free5GC provides a preliminary 3GPP-based implementation in the Docker Compose tool that we used to adapt to Kubernetes.

To start creating our Kubernetes cluster we leverage the well-known Kubernetes environment named Rancher. Rancher, to work properly, needs a host with a Linux operating system and with a Docker daemon installed on it. Docker will be used to support the deployment of the cluster and also the containerization of the core components. Once started, Rancher allows to manage the cluster from a dashboard that allows having a visual representation of it - a much more intuitive approach compared to a deployment managed from the command line. Then, in order to start the deployment of the 5G core in the Kubernetes cluster we need to provide the information of all components to *kubectl*, the Kubernetes command-line tool, in a **.yaml** file. The essential 5G core modules are AMF, AUSF, BSF, HSS, MME, NRF, NSSF, PCF, PCRF, SGWC, SGWU, SMF, UDM, UDR, and UPF. All **.yaml** files contain the configuration of these components and references to dependent components. Also, these files contain settings related to the hosting and communication capabilities inside the cluster. Along with these files very specific to each module, we also have a general **.yaml** file that defines the overall structure of the cluster and is used when performing the deployment of the core so to have a reference to all the modules and their respective **.yaml** files. This file allows to have a central point of configuration of the cluster, making the whole process less complicated and efficient as it reduces all the complexity and time needed to manually change all the files for different 5G core components. Note that for simplicity all 5G core modules are deployed in the same machine. However, Kubernetes allows multi-cluster deployments and management for different site scenarios.

IV. Test Cases and Evaluation

In this section, we evaluate the performance of the 5G core deployment in two different local campus networks that mimic, respectively, an industrial plant and a local area in a smartcity. In particular, we show the performance of failure recovery and scaling of the proposed Kubernetes-based cluster.

We conducted all the experiments in a PC that runs Ubuntu 18.04.6 LTS and is equipped with an Intel Core i5-6500 CPU with 4 cores clocked at 3.2 GHz, and 8 GB of DDR4 memory.

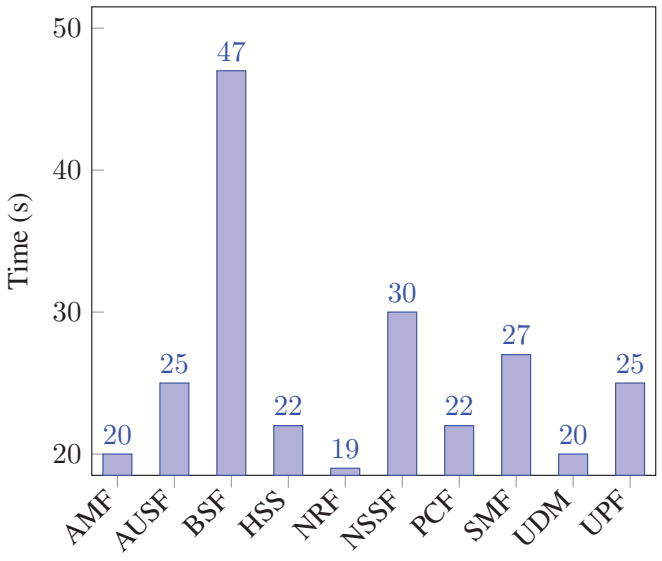

Fig. 2. Failure Recovery Times of the 5G core elements

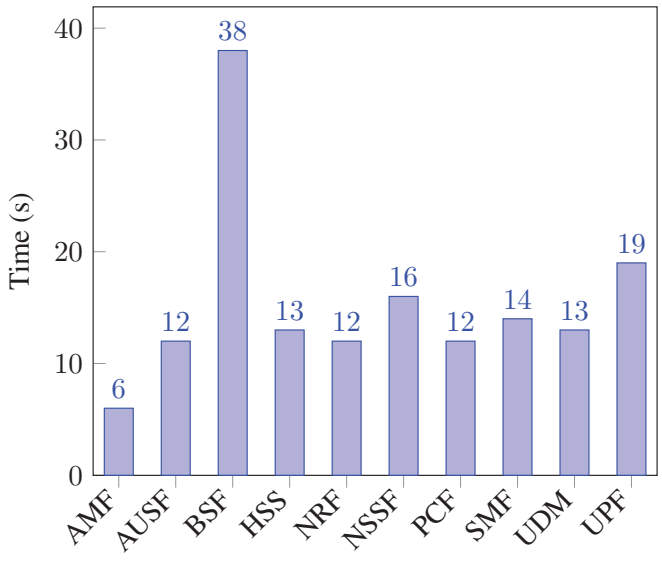

Fig. 3. Scaling Time per Unit of the major 5G core elements

A. *Failure and Scaling*

In the life-cycle of a Pod it may happen that it faces issues resulting in a failure status of the Pod itself. It also may happen that scaling is required to respond to an increase in the volume of requests coming towards the cluster. The goal of this test case is to define how the cluster deals with both situations: a first one where it finds one or more Pods in a failure status; and a second where it requires to perform the scaling up of its components.

The main focus is on the time required to detect a problem within the cluster and then the time needed to recover from the failure. Kubernetes performs a continuous monitoring of the cluster and thanks to this feature we can appreciate how it almost instantly detects the problem and starts the procedure to recover the state of failed Pods. To ensure this, we completely switch off the pods. In terms of the time needed to the cluster to recover from such failures, Fig. 2 shows the measurements for the essential 5G Core elements. The graph shows how, for most of the elements, less than 30 seconds are needed to have a fully running backup, except for the BSF that needs to wait for the other module for dependency issues.

There are many circumstances which may require a scaling up of the cluster components - one significant case in response to high volume of requests coming towards the cluster along with having a good flexibility within the system. A scaling up may also be required only for certain periods of time to respond to an increased necessity of the resources. Figure 3 shows the time (in seconds) needed to perform a scaling-up action per single unit of core components.

We can observe that for obvious reasons the time needed to perform a scaling up is shorter compared to the one needed for failure recovery. The same timing scenario happens here, where the BSF is the only element that represents an exception, since it requires a much higher time than other elements as it has many more dependencies to manage, such as w.r.t. NRF.

B. *Industry 4.0 Scenario*

As stated before, the standard deployment contains all the 5G core modules: it takes around 3 minutes to be deployed in our test environment and consumes around 5 GB of memory. On the contrary, our Industry 4.0-based 5G core deployment includes only a subset of the 5G core modules including AMF, AUSF, NRF, NSSF, PCF, SMF, UDM, UDR, and UPF. Figure 4 shows the measurements of the industrial scenario deployment. This configuration is suitable in a deployment of a static infrastructure like the one needed in an Industry 4.0 scenario requiring very little mobility. This specific test compares the above mentioned deployments in terms of resource consumption and time required to successfully complete the deployment process.

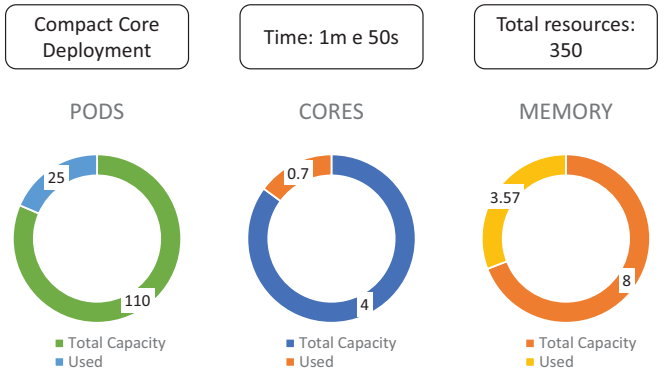

Fig. 4. Industry 4.0-based 5G core Deployment Measurements

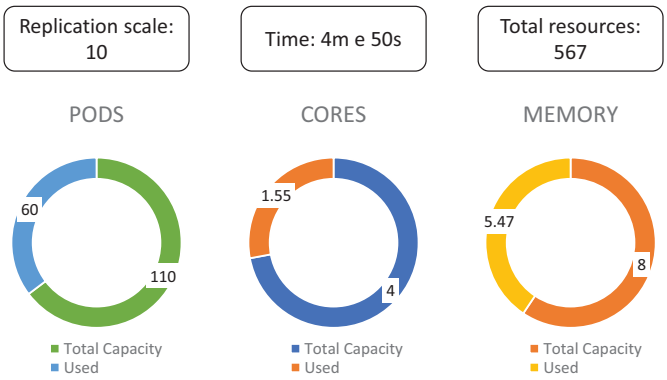

Fig. 5. Smart City-based 5G core Deployment Measurements

When comparing the time required to perform the deployment of the industrial scenario to the one needed for a full version, we can stress that the first case requires nearly half the time. In terms of the resources instead the number of Pods is less compared to the full core deployment but not so low as expected. In reality, this behaviour is understandable as there are several other elements in the background supporting this deployment, which means that the interested number decreases only by the elements not needed in the compact deployment. The Industry 4.0 like infrastructure requires 0.4 units of processing capacity less than the full deployment. A similar behaviour can be observed if we look at the memory - it falls from 5 GB to 3 GB meaning that this second deployment would certainly be useful in scenarios with very less resources at disposal and standard functional requirements.

C. Smart City Scenario

Very different requirements apply to a Smart City scenario where we require high mobility, great throughput and the capacity to manage huge quantities of data. To support these constraints, the network must adapt to a less conventional structure applying a replication degree to key resources so to create a new deployment differentiated from the previous ones in its user plane and control plane configuration.

To enable the mobility and flexibility of the infrastructure some key elements can be scaled up both for control plane and user plane. In this test case, replication has been applied to SMF and AMF in the control plane and to UPF in the user plane. Huge numbers of User Equipment connecting to the core demands a good replication of the above mentioned elements - especially the UPF. The test case was not limited to a re-configuration of the system but also focused on the measurement of resources and time needed for the deployment of the system with different degrees of replication. As a result, we were able to conclude that the cluster has a very satisfactory behaviour when tested with differentiated scenarios and replication factors both in terms of the complexity of re-configurations and resources required to have a functioning system.

Figure 5 shows the resources consumption for a deployment with a replication degree of 10. We have tested the system in various scenarios but we consider this case a good example to represent the behaviour of the cluster stressed with very high replication along with great quantities of incoming requests from a wide range of devices. This test highlights how limited is the difference of resources needed to deploy a system with the above mentioned characteristics compared to a normal cluster with no replication. That means that deployment with great mobility and high throughput can be achieved in a very cost-effective manner.

Notwithstanding these small-scale testbeds, we are confident that the use of Kubernetes as the 5G orchestrator guarantees large-scale deployment with the same performances.

V. Conclusion

To promote the adoption of solutions Kubernetes-based for 5G core deployment, it is crucial to introduce a standard distribution of the 5G core in order to reduce related costs. Thus, the implementation of the 5G core modules as containerized CNFs is

considered fundamental to achieve this goal. It is also vital to facilitate the definition of the CNFs by leveraging DevOps according to the GitOps concept to ensure the management of components' life cycles.

In this work, we have proposed a low-cost ready-to-deploy solution for the 5G core deployment based on Kubernetes available for the community. In particular, this paper is intended to be a tutorial to present a novel research direction in the field of 5G Core Network management. Finally, we tested different deployments of the 5G core in two meaningful scenarios such as Industry 4.0 and Smart City. The results show the costs of the deployments in terms of time and total resource consumption.

For the future, we are thoroughly working on our 5G-Kube. On the one hand, we work on the implementation of the 5G-Kube control plane and we plan to release all the details in the near future. On the other hand, we are now extensively evaluating 5G-Kube in several more significant deployment and real-world testbed scenarios.

REFERENCES

[1] "3GPP TR 21.915 version 15.0.0 Release 15," https://www.etsi.org/deliver/etsi_tr/121900_121999/121915/15.00.00_60/tr_121915v150000p.pdf, [Online; accessed 30-Oct-2022].

[2] T. A. Sisay, D. Scotece, R. Bassoli, D. Barattini, F. Granelli, L. Foschini, and F. Fitzek H. P., "Msn: A playground framework for design and evaluation of microservices-based sdn controller," *Journal of Network and Systems Management*, vol. 30, no. 1, pp. 1573–7705, 2021.

[3] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, dec 2020.

[4] "5G architecture for hybrid and multi-cloud environments," https://www.ericsson.com/4962e4/assets/local/reports-papers/ericsson-technology-review/docs/2022/5g-architecture-for-hybrid-and-multi-cloud-environments.pdf, [Online; accessed 30-Oct-2022].

[5] "Open Source MANO," https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseONE-FINAL.pdf, [Online; accessed 30-Oct-2022].

[6] W.-C. Chang and F. Lin, "Coordinated management of 5g core slices by mano and oss/bss," *Journal of Computer and Communications*, vol. 09, pp. 52–72, 01 2021.

[7] F. Beetz and S. Harrer, "Gitops: The evolution of devops?" *IEEE Software*, vol. 39, no. 4, pp. 70–75, 2022.

[8] "System Architecture for the 5G System," https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.03.00_60/ts_123501v150300p.pdf, [Online; accessed 30-Oct-2022].
099/011/02.06.01_60/gs_nfv-ifa011v020601p.pdf, [Online; accessed 30-Oct-2022].

[10] W. Shen, M. Yoshida, K. Minato, and W. Imajuku, "vconductor: An enabler for achieving virtual network integration as a service," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 116–124, 2015.

[11] H. T. Nguyen, T. Van Do, and C. Rotter, "Scaling upf instances in 5g/6g core with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 165 892–165 906, 2021.

[12] "Cloud Native and 5G Verticals' services," https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-SN-WG-5G-and-Cloud-Native.pdf, [Online; accessed 30-Oct-2022].

[13] "Open Source implementation for 5G Core and EPC," https://open5gs.org/, [Online; accessed 30-Oct-2022].

[14] "free5GC," https://www.free5gc.org/, [Online; accessed 30-Oct-2022].

[15] "Cloud Native Thinkin," https://github.com/cncf/telecom-user-group/blob/master/whitepaper/CNCF%20_TUG_Cloud%20Native%20Thinking%20for%20Telecommunications.pdf, [Online; accessed 30-Oct-2022].

Domenico Scotece is a postdoc researcher at the Univ of Bologna. He received his Ph.D. and M.Sc. degrees in Computer Science Eng. from Univ of Bologna in 2020 and 2014, respectivly. His research interests include pervasive computing, middleware for fog and edge computing, software-defined networking, IoT, and management of cloud computing systems.

Asma Noor is a senior software engineer and she completed her M.Sc. degree in Computer Science Engineering from the university of Bologna in 2022. Her interests include distributed systems, cloud computing and software engineering of complex industrial software solutions.

Luca Foschini graduated from the University of Bologna, where he received a Ph.D. degree in computer science engineering in 2007. He is now an associate professor of computer engineering at the University of Bologna. His interests span from integrated management of distributed systems and services to wireless pervasive computing and scalable context data distribution infrastructures and context-aware services. Currently, he is working on mobile crowdsensing and crowdsourcing and management of cloud systems for smart city environments.

Antonio Corradi (Life Senior Member, IEEE) received the Graduate degree from the University of Bologna, Italy, and the M.S. degree in electrical engineering from Cornell University, USA. He is currently a Full Professor in computer engineering with the University of Bologna. His research interests include distributed systems, middleware for pervasive and heterogeneous computing, infrastructure for services and network management, and cloud continuum and multicloud.