Research on 5GC Cloud-Native Deployment Mechanism Based on Cloud-Edge Collaboration

Yijun Cao
China Telecom Corporation Limited Zhejiang Branch
Hangzhou, Zhejiang, China
caoyj.zj@chinatelecom.cn

Peng Lv
China Telecom Corporation Limited Zhejiang Branch
Hangzhou, Zhejiang, China
lvp.zj@chinatelecom.cn

Xiaojie Zhou
China Telecom Corporation Limited Zhejiang Branch
Hangzhou, Zhejiang, China
zhouxj.xc.zj@chinatelecom.cn

Yuting Wu
China Telecom Research Institute
Beijing, China
wuyt6@chinatelecom.cn

Shuo Quan
China Telecom Research Institute
Beijing, China
quansh@chinatelecom.cn

Zeya Zhu
China Telecom Research Institute
Beijing, China
zhuzy13@chinatelecom.cn

Shen Gao*
China Telecom Research Institute
Beijing, China
gaos13@chinatelecom.cn

2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS) | 978-1-6654-7053-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/TOCS56154.2022.10016065

***Abstract*—With the proliferation of 5G business scenarios, 5G core network (5GC) cloudification has a major trend for both operators and vendors. As a novel cloud computing technology, cloud native can realize agile development, rapid deployment, on-demand customization of resources, and flexible business innovation of 5GC. In order to promote the cloud native evolution of 5GC, this paper proposes a prototype of cloud native 5GC platform based on open source projects, and conducts deployment experiment. However, open source projects have shortcomings in the deployment of cloud native 5GC, such as being unable to remote user plane function (UPF) to edge cloud, not supporting multiple network planes, and relying on static network function IP addresses. From the conclusion of experiment, a set of cloud-edge collaboration 5GC deployment solutions are accomplished, proving cloud native technology can carry the services of 5GC.**

Keywords—5G core (5GC), Kubernetes, KubeEdge, Cloud Native

I. INTRODUCTION

The major mission of 5th generation mobile networks (5G) is not only the communication among people, but also to provide faster, more reliable and more flexible network services for all walks of society. The International Telecommunication Union (ITU) has defined three major application scenarios for 5G [1], including enhanced mobile broadband (eMBB), massive machine type communication (mMTC) and low latency and highly reliable communication (uRLLC).

To satisfy the needs of 5G networks to flexibly provide a variety of network services for different business scenarios, the 5G core network (5GC) introduces a service-based architecture (SBA). 3GPP TS 23.501 proposes a service-based 5GC architecture [2], which splits the network function (NF), and the split NFs communicate through a service-based interface (SBI). The introduction of SBA enables 5GC to have open network capabilities, flexible expansion capabilities, and reliable fault handling capabilities. At the same time, 5GC also introduced Network Function Virtualization (NFV) technology, so that each NF in 5GC can get rid of the constraints of traditional proprietary hardware and be deployed on standard servers. With the development of NFV technology, each NF in 5GC can be deployed and run-on virtual machines. However, virtual machines have dependencies on operating systems, applications, and libraries, and are not highly portable.

The emergence of cloud-native technologies has injected new vitality into the development of 5GC. In 2017, the European Telecommunications Standards Institute (ETSI) released the first standard for cloud-native network functions, NFV EVE011. This standard defines the non-functional characteristics of virtualized network functions (VNFs) based on cloud-native technologies [3], such as elastic scaling, robustness, containerized deployment, automated management, declarative APIs, etc. These features of cloud native VNFs complement the deficiencies in performance and scalability brought by virtual machine deployment. In 2018, S. Imadali and A. Bousselmi proposed a design model of unified cloud-native software platform - 5GaaS [4], which can deploy operator network resources, mobile communications, and cloud computing services. In 2020, O. Arouk and N. Nikaein proposed a cloud-native 5GC service platform, which can support the deployment of cloud-native 5G applications [5]. In the same year, they also proposed an implementation scheme for cloud-native deployment of 5G networks [6], which uses Kubernetes as the orchestration system, uses OpenShift for service governance, and implements network deployment through containerized OpenAirInterface (OAI) to complete the cloud-native deployment of 5G networks.

In the past two years, research papers on the cloud-native deployment of 5GC have continued to emerge. A. Khichane proposed a lifecycle management and automated scaling method for different business scenarios in cloud-native 5GC [7]. Á. Leiter proposed a fault handling method for cloud-native user plane network function UPF) [8]. Among them, it is worth noting that, T. Tsourdinis proposed a method to use multi-access edge computing (MEC) technology to deploy network functions that migrate to the user side and reduce cloud-native 5GC communication latency [9]. Álvaro also proposed and implemented a cloud-edge collaboration 5GC service platform based on cloud-native technology [10]. In addition, in the 5G private network scenario, some network functions need to be deployed on the user side. It is also

necessary to introduce MEC technology to build network functions which deployed on edge cloud. Therefore, the deployment scheme of cloud-edge collaboration will become one of the 5GC deployment schemes that have received a lot of attention and application.

This paper proposes a containerized 5GC deployment solution for cloud-edge collaboration. This paper uses Free5GC to implement 5GC functional services, Kubernetes to manage and orchestrate containerized network functions, and KubeEdge to implement UPF remote to the user side. The deployment method proposed in this paper solves the common technical bottlenecks encountered in the 5GC cloud-side collaborative deployment process, such as the UPF remote, the 5GC network functions containerization, the multi-network plane, and network functions providing services to the external network.

II. Overview of Cloud Native 5GC

A. 5GC Architecture and Free5GC

3GPP TS 23.501 proposes two 5GC architectures, which are based on service and based on reference point [2]. Fig. 1 shows the 5GC architecture based on service.

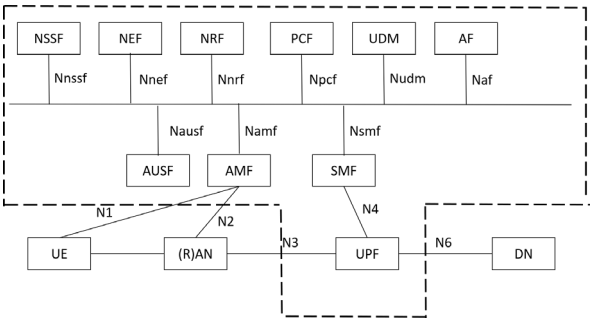

Fig. 1. Service based 5GC architecture

The service-based 5GC architecture is divided into two parts: the control plane and the user plane. The control plane includes network functions for management, such as Access and Mobility Management Function (AMF), Session Management Function (SMF), Authentication Service Function (AUSF), and Unified Data Management Function (UDM). The control plane network functions provide external services through the SBI based on the HTTP2 protocol. The SBI of the network function is Namf, Nsmf, Nausf, etc. in Fig. 1. The 5GC control plane implements business functions such as access management, data storage, user authentication, slice management, and policy management. The user plane is responsible for forwarding user data, the main network function is UPF, and the protocol used for data forwarding is the GTP-U protocol. The main interface for communication between the control plane and the user plane is the N4 interface. The SMF uses the PFCP protocol to issue various session requests to the UPF to control the data forwarding of the UPF.

As one of the current mainstream 5GC open-source projects, free5GC implements the main network functions, database, and user interface of 5GC based on the 3GPP R15 standard. In addition to supporting the direct deployment of network functions and applications in the form of processes and running on virtual machines, Free5GC also supports the use of Docker Compose to deploy various components of 5GC in containers. The main technical route of using Docker Compose for 5GC containerized deployment includes the construction of each network function image and the use of IPAM for network configuration. This paper draws on the mirror construction part of this solution, and on this basis, splits the network function mirroring function in a more fine-grained manner.

B. Cloud Native Edge Computing and KubeEdge

To implement remote deployment of UPF near the user side, it needs to support lightweight deployment, high reliability, high security, large-scale management, and unified scheduling. In the case of unstable edge network, edge autonomy and data consistency guarantee can be performed during network disconnection and recovery. Based on the above requirements and cooperating with Kubernetes to manage deployment, this paper uses the cloud-native edge computing platform to implement the deployment of UPF sinking to the network edge.

Currently, the more mature cloud-native edge computing platforms include Huawei's open-source KubeEdge, Rancher's open-source K3s, Alibaba's open-source Openyurt, and Tencent's open-source SuperEdge. This paper compares and analyzes the key capabilities required by the above open-source computing platforms to carry UPF, as shown in Table I.

TABLE I. Edge Computing Platforms

	KubeEdge	**K3s**	**OpenYurt**	**SuperEdge**
CNCF stage	Incubating	Sandbox	Sandbox	Not CNCF project
Time of open-source	2018.11	2019.2	2020.5	2020.12
Lightweight	Yes	Yes	No	No
Cloud-edge collaboration	Yes	Rely on multi-node manage	Yes	Yes
Edge autonomy	Yes	Yes	Yes	Yes
Stability	Too much user equipment	Low system stability	Large scale node & Long time recovery	Large scale node & Long time recovery
Compatibility of cloud native	Partially compatible	Compatible	Compatible	Compatible
Monitoring	Partially supported	Supported	Supported	Supported
Device management	Yes	No	No	No

Among them, KubeEdge has the highest degree of usage and maturity, OpenYurt/SuperEdge's architecture design is more elegant, and K3s is derivative from Kubernetes. According to the bearing requirements of UPF and the analysis of technology maturity, this paper chooses KubeEdge as the edge computing platform.

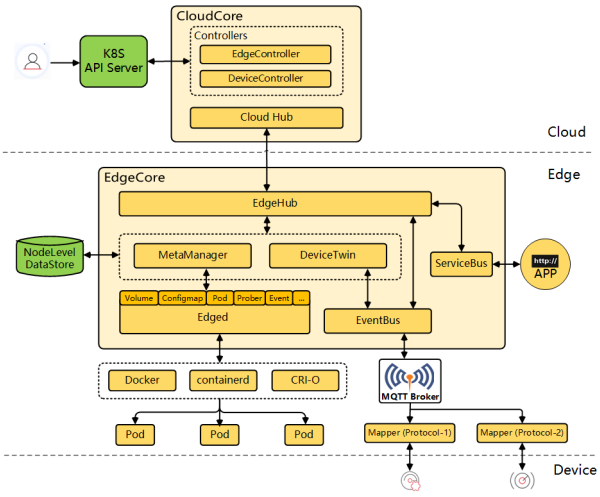

Fig. 2. KubeEdge architecture

The architecture of KubeEdge is shown in Fig. 2., which is divided into CloudCore and EdgeCore. CloudCore is deployed in the cloud. Controllers are the controllers extended by Kubernetes, which are used to manage edge nodes, edge devices, data cloud-side synchronization, etc. CloudHub is a Web Socket server that monitors changes in the cloud and communicates with EdgeCore's Web Socket client EdgeHub. EdgeCore is deployed at the edge, Edged is an agent running on the edge node to manage containerized applications, EventBus is an MQTT client that provides subscription and publishing functions for components, and ServiceBus is an HTTP client that provides ability of cloud services access the edge HTTP server, MetaManager is responsible for message processing, and DeviceTwin is responsible for device state storage and synchronization.

III. Experimental Environment

A. Experimental Topology

The experimental topology is shown in Fig. 3. We use a total of three physical nodes, two of which are used to simulate the cloud environment and deploy Kubernetes clusters, and the other node is used to simulate the edge environment and deploy KubeEdge components. As a result, we have built a cloud-native infrastructure platform with cloud-side collaboration capabilities. On this platform, we can deploy 5GC-related network functions. The 5GC control plane network functions, such as AMF, SMF, AUSF, etc., are deployed in the Kubernetes cluster located in the central cloud, while the 5GC user plane network functions UPF are deployed in the KubeEdge node through scheduling.

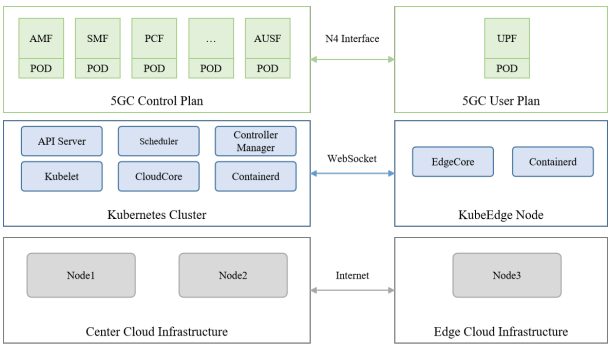

Fig. 3. Experimental topology

In terms of network connectivity, basic IP network reachability is required between the central node and the edge node. The Kubernetes cluster in the central cloud and the KubeEdge in the edge cloud communicate through Websocket, while the 5GC control plane network functions and user plane communication using the N4 interface. The experimental environment meets the above communication requirements through the local area network.

B. Hardware Setup

The experimental environment of this paper uses 3 DELL R740 servers, and the main configuration of each server is:

- 2 x Intel Xeon Silver 4214R 2.4GHz
- 256GB RAM
- 2 x 4TB disk space
- 2 x 10G network interface card

This article uses Huawei CloudEngine S5731-H optoelectronic hybrid switch to implement network communication between the three servers. The configuration of switch is:

- 24 x 10/100/1000Base-T Ethernet ports, 4 x 10GE SFP+ ports
- One extended slot
- Switching capacity: 288 Gbps/672 Gbps

C. Software Setting

We installs Ubuntu 18.04 operating system on three servers with 5.4.0-65-generic kernel version. The GTP5G module is installed upon the operating system kernel to support the forwarding of GTP-U protocol data. The software configuration on the operating system is shown in Table II.

TABLE II. SOFTWARE CONFIGURATION

Configuration item	Configuration information
Compiling Environment	• gcc v7.3.0 • Golang v1.14.4 linux/amd64
Database	MangoDB v4.4
SDK	git, cmake, autoconf, libtool, pkg-config, libmnl-dev, libyaml-dev
Network Configuration	• ipv4 forwarding • DNAT for data network • turn off firewall
Management and Orchestration System	Kubernetes v1.25.0
Container Registry	Harbor v2.6.0
Egde Cloud	KubeEdge v1.11.1
5GC Network Functions	free5GC v3.2.1

IV. Solution Design and Implementation

A. Deployment of UPF on Edge Cloud

The UPF is responsible for the routing and forwarding functions of the user plane data packets of the 5G core network. All data must be forwarded by the UPF before it can flow to the external network. Therefore, deploying UPF to the edge of the network can reduce the transmission delay, realize the local shunting of data traffic, relieve the data.

Transmission pressure on the core network. Additionally, it can improve the network data processing efficiency, and satisfy the requirements of vertical industries for ultra-low-latency, ultra-high-bandwidth, and security. The deployment of 5GC based on KubeEdge is shown in Fig. 4.

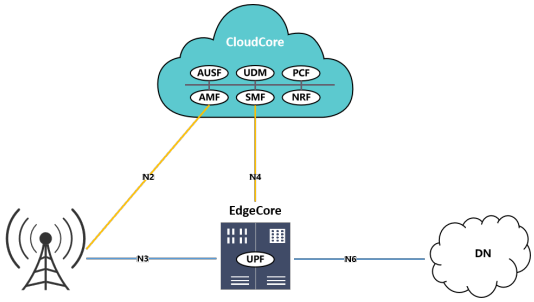

Fig. 4. Deployment of 5GC based on KubeEdge

B. Constitution of Image

The core of cloud native technology is containerization, and the first step of containerization is to build an image. In 5GC, components that require containerized deployment can be divided into two categories. One category is general-purpose applications, such as databases, monitoring systems, etc. These components are usually used by multiple or all components. The other type is network functions, such as AMF, UDM, etc. These components are used to complete specific services in 5GC. The image constitution methods for these two types of components are different. For general-purpose applications, images can usually be found in publicimage repositories, which can be downloaded and used directly. For network functions, it is necessary to logically split the construction objects and define appropriate operations for each layer of the mirror according to the 5GC business requirements. Fig. 5. shows construction method of the multi-layer image of the 5GC network function.

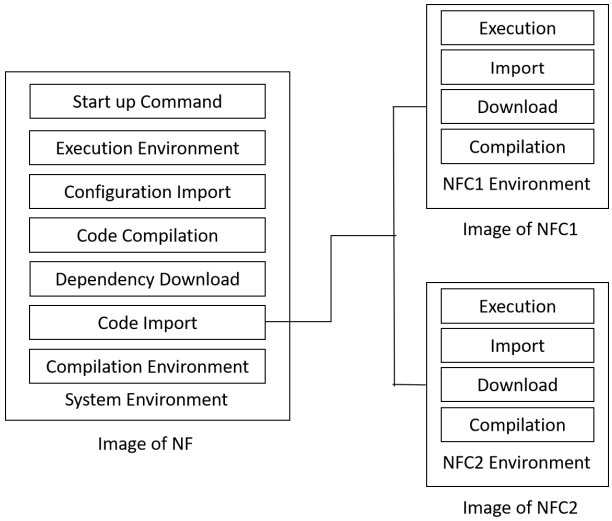

Fig. 5. Image constitution of 5GC network function

The left side of Fig. 5. shows the image constitution procedure of a network function from environment configuration to process startup, which can implement a network function running in a container. If the network function needs to be further divided, the source code of the network function needs to be divided into components that can run independently and communicate with each other after compilation before code compilation. In fact, in the process of containerized construction of network functions, the internal logic splitting of a single network function has always been a topic worthy of study. Among them, the schemes that can be referred to include splitting by network function service, splitting by the source code class, and splitting by the state of services.

C. Realization of Multiple Network Plane

The 5GC network functions require multiple network planes to realize the separation of control plane and data plane. For example, UPF needs at least four network interfaces are required, one is connected to SMF (N4) to implement network controlling of UPF, two are connected to (R)AN and DN (N3 and N6) to implement data transmission, and one is connected to the management and orchestration system. However, Kubernetes and KubeEdge do not support multiple network planes. When creating a Pod, only one network interface can be created for a workload. Therefore, this paper uses the Multus CNI plug-in to realize the creation of multiple network planes of 5GC network functions. The planning and deployment of the network plane is shown in Fig. 6. Multus CNI is responsible for creating multiple network cards for containerized 5GC network functions, and CNI plug-ins such as Calico, Flannel, MacVlan, and SRIOV are responsible for creating specific network cards, including creating virtual network devices.

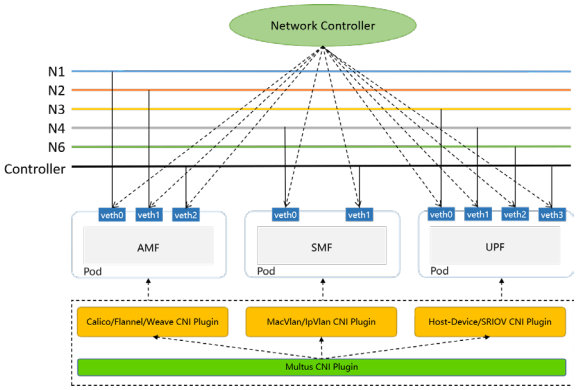

Fig. 6. Deployment 5GC multiple network plane

The network controller configures the network functions and attaches the NICs to the corresponding network planes.

D. Implementation of Static IP

During the startup configuration of Free5GC's control plane network functions, there will be situations where a static IP needs to be written into the configuration file. For example, when configuring the AMF, it is necessary to write the static IP of the AMF SBI and the static IP of the NRF to be registered. In fact, for business requirements, it is also a reasonable solution to assign a static IP to the service interface of the containerized network function. Otherwise, the configuration file needs to be continuously modified during the process of container restart and capacity expansion. This paper uses OpenvSwitch to create a virtualized bridge br, and then uses the CNI network controller in the Docker public image repository to configure the IP for the network function and mount it to the br, which realizes the static IP of the service interface of the network function. The configuration is shown in Fig.7.

The red box in Fig.7. shows the core steps of configuring static IP. By deploying the network controller in the Init container and using the network configuration commands compatible with the client program in the CNI network controller, the static IP configuration of the specific network card in the network function container is realized, and the configured network interface can be regarded as a network function to provide the SBI interface to other components.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nf-deployment
spec:
  selector:
    matchLabors:
      app: nf
  .......
  initContainers:
  - name: network-controller
    image: sdnvortex/network-controller:v0.4.9
    command: ["/go/bin/client"]
    args: ["-s=unix:///tmp/vortex.sock", "-b=br", "-
n=eth1", "-i=192.168.1.1/24"]
  containers:
   - name: network-function
     image: free5gc-nf
     .......
```

Fig. 7. Configuration of static IP address

E. Acceleration for KubeEdge

UPF is deployed in a virtual machine container mode, and the forwarding of service traffic needs to pass through multiple data such as physical network card, host kernel network protocol stack, kernel state virtual switching layer, virtual machine kernel state network protocol stack, and virtual machine user state UPF network function. In the forwarding channel, there are many system interruptions, context switching, memory copying, and virtualization and decapsulation operations, which cannot meet the requirements of UPF as a forwarding network function for high-performance network forwarding capabilities. This paper uses two methods to enhance the network performance of UPF. One is the para-virtualization method of OVS-DPDK plus Virtio, which bypasses the kernel protocol stack for performance optimization and uses the MacVlan CNI plug-in; the other is to use the SRIOV method to VF is directly used by UPF, using the Host-Device CNI plug-in (SRIOV CNI can be used in bare metal containers), as shown in Fig. 8. The first para-virtualization method is not limited by the number of VF network interfaces, and the east-west traffic performance is better, but the overall performance is not as good as the second method of pass-through.

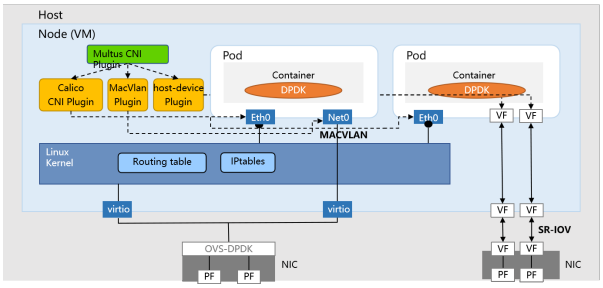

Fig. 8. High performance network of KubeEdge

V. Conclusion

This paper proposes and implements a cloud-edge collaboration 5G core network deployment scheme based on the open-source project free5GC. The cloud side adopts cloud native technology and uses Kubernetes as the orchestration management system to realize the containerized deployment of the 5G core network. On the edge side, the open-source project KubeEdge is used to realize the remoting of the UPF of the user plane network function. In addition, this paper also solves the technical bottlenecks encountered in the deployment process, such as pod multi-network plane and network function service interface static IP. We introduce high-performance network plug-ins on the edge side to improve the forwarding performance of data. Deployment experiments show that the cloud-edge collaboration scheme proposed in this paper can realize various business scenarios in 5GC and enhance the user plane network performance of 5GC. Besides, key indicators such as elastic expansion, network slicing, and reliability of 5GC need to be further deployed and verified in future work.

VI. Future Work

The use of Kubernetes and KubeEdge is for IT systems, lacking the routing and management of telecom proprietary network protocols, such as SCTP, SIP, GTP, etc., using Kubernetes and KubeEdge to carry 5GC network functions, management and observability of network traffic, telecom proprietary protocol load balancing and other capabilities need to be enhanced.

References

[1] ITU-R M.2083-0, Framework and overall objectives of the future development of IMT for 2020 and beyond[S]. Geneva: Radiocommunication Sector of ITU, 2015.

[2] 3GPP TS23.501 V17.5.0, System architecture for the 5G System[S]. Valbonne: 3rd Generation Partnership Project(3GPP), 2022.

[3] ETSI GS NFV-EVE 011 V3.1.1, Network Functions Virtualisation (NFV) Release 3; Virtualised Network Function; Specification of the Classification of Cloud Native VNF implementations[S]. Cedex: European Telecommunications Standards Institute (ETSI), 2018.

[4] S. Imadali and A. Bousselmi, Cloud Native 5G Virtual Network Functions: Design Principles and Use Cases, 2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2), 2018, pp. 91-96, doi: 10.1109/SC2.2018.00019.

[5] O. Arouk and N. Nikaein, Kube5G: A Cloud-Native 5G Service Platform, GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9348073.

[6] O. Arouk and N. Nikaein, 5G Cloud-Native: Network Management & Automation, NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1-2, doi: 10.1109/NOMS47738.2020.9110392.

[7] A. Khichane, I. Fajjari, N. Aitsaadi and M. Gueroui, Cloud Native 5G: an Efficient Orchestration of Cloud Native 5G System, NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, 2022, pp. 1-9, doi: 10.1109/NOMS54207.2022.9789856.

[8] Á. Leiter, A. Hegyi, N. Galambosi, E. Lami and P. Fazekas, Automatic failover of 5G container-based User Plane Function by ONAP closed-loop orchestration, NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, 2022, pp. 1-2, doi: 10.1109/NOMS54207.2022.9789799.

[9] T. Tsourdinis, N. Makris and T. Korakis, Experimental evaluation of a Follow-me MEC Cloud-Native 5G network, 2021 IEEE 4th 5G World Forum (5GWF), 2021, pp. 394-399, doi: 10.1109/5GWF52925.2021.00076.

[10] Á. Vázquez-Rodríguez, C. Giraldo-Rodríguez and D. Chaves-Diéguez, A Cloud-Native Platform for 5G Experimentation, 2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2022, pp. 60-64, doi: 10.1109/BlackSeaCom54372.2022.9858299.

[11] Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.