A Cloud-Native Platform for 5G Experimentation

Álvaro Vázquez-Rodríguez, Carlos Giraldo-Rodríguez, David Chaves-Diéguez
GRADIANT (Galician Research and Development Center for Advanced Telecommunications), Vigo, Galicia, Spain
Email: {avrodriguez, cgiraldo, dchaves}@gradiant.org

2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom) | 978-1-6654-9749-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/BLACKSEACOM54372.2022.9858299

***Abstract*—Nowadays, with the natural evolution towards cloud-native technologies due to the needs of current 5G and future mobile network deployments, there is a need for platforms that bring together the necessary features to facilitate development and research in this field. It is necessary to have these infrastructures for the operation of interconnected services, automating the network service lifecycle, supporting real time restrictions and the coexistence of physical and virtual functions. Concretely, the presented platform will be used to experiment with 5G cloud-native network services in a flexible, agile and lightweight way. This paper presents a 5G cloud-native platform using open source components, fully functional and valid for experimental or proof-of-concept deployments.**

***Index Terms*—5G, Cloud-Native, experimentation, network virtualization, orchestration**

I. Introduction

In the last decade we have witnessed a true Network Softwarization process, first with Software Defined Network (SDN) followed by Network Function Virtualization (NFV), which helped network researchers to experiment leveraging virtualization platforms such as OpenStack or VMWare. Additionally, IT virtualization is transitioning from Virtual Machines to more lightweight and dynamic approaches in the form of containers. Virtual network Functions (VNFs) are also evolving into their containerized counterpart, the Cloud-Native Network Functions (CNFs). These tendencies lead to the design of 5G testbeds that combine the deployment of the necessary services of 5G networks with the advances in cloud-native IT Technologies.

Following the ideas of [1], a Platform-as-a-Service (PaaS) for 5G experimentation is proposed in this work. PaaS is conceived for a cloud environment, allowing us to manage the entire lifecycle of a 5G experiment, from its design, deployment and support thanks to established and fully functional DevOps techniques. This new approach allows greater flexibility and scalability. With the evolution of VNFs towards CNFs, packaging network code as containers is only the beginning of the work to be done. The number of containers that can compose a network application and all the complex requirements such as load balancing, monitoring, failover and scalability need to be addressed and orchestrated in this new PaaS approach.

There are alternatives to orchestrate these platforms: OpenSourceMano (OSM) is an NFV orchestration platform implementing the ETSI NFV standards. Kubernetes is the de-facto standard to orchestrate Containers and there is a strong open-source community backed by the Cloud Native Computing Foundation (CNCF), with multiple projects tools. Therefore, a combination of the two is proposed to implement the next stage in 5G experimentation infrastructures.

In this paper, we propose a 5G Experimentation Platform based on the latest cloud-native technologies to help researchers experiment with latest 5G technologies in a dynamic and easy to use way. First, a search of related work and cloud-native tools is shown. Secondly, the proposal is described with its main features and a brief description of its high-level architecture, naming the tools used. Thirdly, the tests carried out to validate the proposal are shown and, finally, some brief conclusions are given and the lines to be followed for future work.

II. Related Work

There are several proposals of PaaS for Mobile Networks. In [2], the authors address how to scale only the Mobility Management Entity (MME) of an LTE network based on the number of connected users using a cloud-native approach. The work of [3] shows how to build a 5G mobile network using OpenAirInterface [11], but without giving a cloud-native approach to building and automating the lifecycle of 5G services. Continuing with the RAN (Radio Access Network) and adding another feature of this platform such as the support of MEC (Mobile Edge Computing) applications, in [4] we see how an architecture is presented using DevOps techniques to automate 5G MEC applications. Compared to this proposal, our proposal adds cloud-native principles and support to automation of the different SDRs to this type of applications, again allowing the complete management of their lifecycle. The authors in [5] propose a platform for deploying 5G CNFs applications by comparing the performance between baremetal, Docker and their platform. The proposal in [6] is a federated platform between different testbeds where the performance of the 5G core is tested through fully containerised services, facilitating the deployment of MEC applications and giving the possibility to experiment through AIOps tools. Another 5G platform in a cloud-native environment is proposed in [7] where the deployment of different services is investigated from a control plane and data plane separation perspective. Finally, regarding a key point in 5G such as Network Slicing, [8] studies the performance of different deployments of static and dynamic slices using NFVs through OSM in a single cluster.

Our 5G CNF platform proposal focuses on experimentation and provides new features to enable a wide range of experiments. The first key point is the separation of the data plane and the control plane for the different services through the use

of Multus [9] or the capacity of Kubernetes to expose services to outside networks. The second key point is the ability to offer, on a single site, a wide range of heterogeneous infrastructure, such as different computing architectures, giving the possibility to perform multi-architecture experiments. Another key point available is the use of multiple orchestrators, allowing the option to emulate cloud-edge environments by setting up different network capabilities such as delay or bandwidth between the orchestrators. These options enable experimenting with key 5G features such as Network Slicing, as described in section IV.

A. Cloud-Native Platforms and Tools

Our experimentation leverages next generation of virtualization (Containers) and orchestration (Kubernetes) technologies under the name "Cloud-Native". Kubernetes is an open-source project for container orchestration system for automating the deployment, scale and management of applications. It performs the functions of virtual infrastructure manager (VIM) and VNF manager (VNFM) following the ETSI's NFV architecture language. The extensibility of Kubernetes has enabled a gradual transition from legacy VNFs built on to a Kubernetes stack. For example the CNF Testbed [10] is an open source initiative that aims to provide reference code and test cases for running the same network code packaged as containers (CNFs) in Kubernetes and as VNFs in OpenStack. Another point of extensibility in Kubernetes uses CNI (Container Network Interface), an open specification of the Cloud Native Computing Foundation (CNCF) to enable different plugins to configure network interfaces in Linux containers.

For the development and deployment of CNFs, a number of tools are available to facilitate this task:

- **The Cloud Native Application Bundle (CNAB)** is a standard packaging format for multi-component distributed applications. It allows packages to target different runtimes and architectures. It empowers application distributors to package applications for deployment on a wide variety of cloud platforms, providers, and services. Furthermore, it provides necessary capabilities for delivering multi-container applications in disconnected environments.
- **Helm** is a cloud-native tool for automating the creation, packaging, configuration, and deployment of applications and services to Kubernetes clusters. Helm deploys packaged applications to Kubernetes and structures them into charts. The charts contain all pre-configured application resources along with all the versions into one easily manageable package.

III. A CNF PLATFORM FOR EXPERIMENTATION

Our proposal [Fig. 1] is a CNF Platform for Experimentation based on open source technologies and aligned with the CNCF. Its main features are:

- Multi-cluster design.
- Support to multiple architectures: amd64 or arm64.
- Multi-network solution.
- Programmable network QoS.
- Support to multiple tenants.
- Support to radio resources (USRPs).

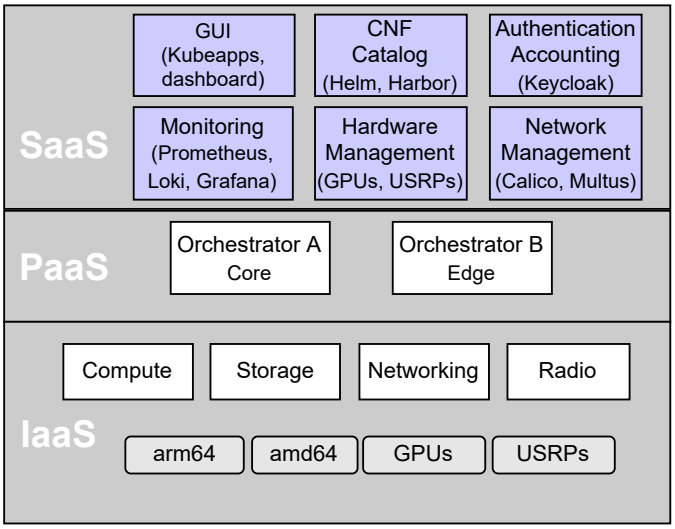

Fig. 1. Architecture

In this section, all the tools introduced in the previous section are put together to create a cloud-native Platform for Experimentation. The design consists of multiple Kubernetes clusters: one for deploying on amd64 compute infrastructure and the other for deploying on both amd64 and arm64 nodes, having the possibility to compare services running on different architectures and any possible combinations. Each cluster has isolated networks for their services using Calico as CNI, but communication between them is achieved through the use of Multus and the capability of Kubernetes to expose services with an external IP. The interconnection of each cluster is done through an OPNsense device allowing the emulation of network QoS with the addition of artificial delays and/or limiting the throughput. This programmable network QoS allows the experimenter to mimic scenarios with loads deployed at the core and the edge.

The cloud-native Platform can be used for different purposes and by different tenants. Using Keycloak as an Authentication Service, tenants can access in a safe and secure way, isolating their workloads from each other. Integrating Keycloak with Kubernetes allows the infrastructure administrator to limit resource usage applying quotes to each tenant.

With respect to the CNF Catalog container images are built from different open source solutions (e.g., open5GS, free5GC, UERANSIM, OpenAirInterface, etc.), and they are built supporting multiple computing architectures. The image sources are publicly available at [12]. These images are used to create packages for deploying in the platform. The packages follow CNCF approaches like Helm or CNAB, enabling the customization of the deployments through their configuration files. The package sources are publicly available at [13]. The platform also offers a private repository with

Harbor, an open source project, where each tenant can upload, upgrade and access their own packages in a secure way.

Kubeapps is used as an application dashboard for tenants, automating the lifecycle of the different services deployed and Kubernetes-dashboard provides the status of the running resources in the Kubernetes clusters. To monitor the workloads and services, similar approaches are used. Prometheus extracts cluster and service data, Loki extracts logs from services and Grafana enables advanced dashboards to show monitoring data through graphs.

To support machine learning and radio access services, the 5G cloud-native Platform also supports dedicated hardware such as Graphics Processing Unit (GPUs) and Universal Software Radio Peripheral (USRPs), with the capability of allocating this hardware in a dynamic manner. Services requesting this hardware will be automatically granted access if hardware is available, or queued until hardware becomes available. In the case of USRPs, this has been done with our own implementation of a Kubernetes device plugin for Ettus devices. The implementation is publicly available at [14].

IV. Platform Validation

In this section we propose a series of experiments to demonstrate some of the platform features. First of all, multiple underlying networks are configured to be able to experiment with the split of control and data planes. Each Kubernetes cluster has two physical networks with different capacities that will be used for control and data traffic respectively. The control plane network has 1Gbps links while the data plane networks has 10Gbps links. Iperf tests were performed to validate the capacity of each network with a summary of the results in table I.

Network	Speed (Gbps)
Control	0.921
Data	9.9

TABLE I
MEASURED BANDWIDTH OF THE UNDERLYING PHYSICAL NETWORKS.

To demonstrate the performance and capabilities of our 5G infrastructure, two tests were planned that use a 5G core (open5GS) and a virtualised gNodeB and UE (UERANSIM) with open source applications which have been packaged as CNFs. The applications are available at [13].

The first experiment validates a deployment that relies on the programmable network QoS, configuring a delay between the Kubernetes clusters of 0ms, 50ms and 100ms to represent communication latency between the cloud and the edge. We deployed the Access and Mobility Management Function (AMF) service of 5G core in the cloud cluster and two gNodeB in both the cloud and the edge cluster [Fig. 2].

After the deployment, round trip time measurements between the gNodeBs and the AMF containers where conducted. The required time to register each gNodeB with the 5G Core was also measured. Figure 3 shows the impact of the latency between edge and cloud in the required time to register the gNodeB. This experiment demonstrates and highlights the utility of the multi-cluster and the programmable network QoS features of our cloud-native platform.

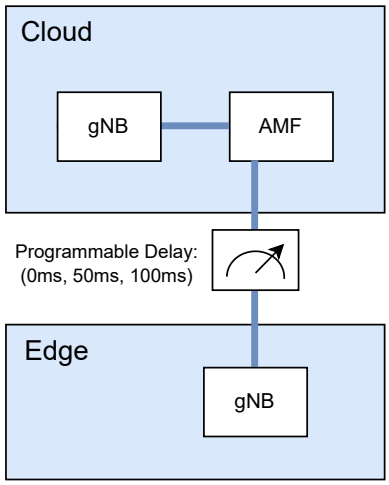

Fig. 2. Components used for test 1.

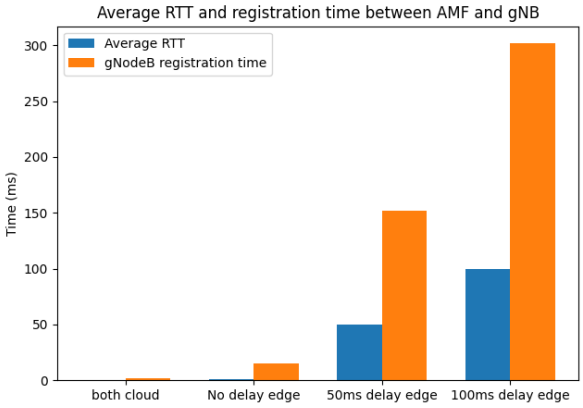

Fig. 3. Average RTT and registration time between AMF and gNB.

The second experiment, more complete than the previous one, uses the multi-cluster, multi-network and the programmable network QoS features to experiment with control and data plane separation and network slicing as shown in Fig. 4. The experiment deploys all functions of the 5G SA core at the cloud cluster with a second User Plane Function (UPF) at the edge cluster. Then, a gNodeB and two User Equipments (UEs) are deployed at the edge clusters. Again, a delay of 0ms, 50ms and 100ms are configured between the core and edge clusters.

Two network slices are configured at the core: the first slice with the UPF at the core and the second slice with the UPF at the edge. Finally, each UE is associated with a different slice for Internet access.

Round trip time measurements between the UEs and a public internet server (concretely, Google's DNS server 8.8.8.8)

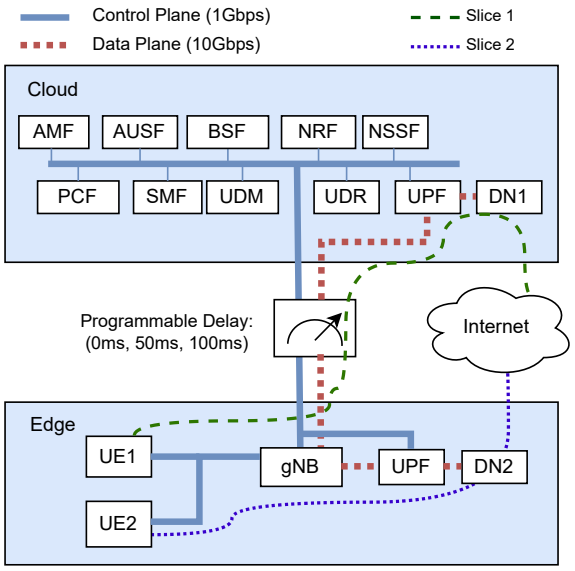

Fig. 4. Components used for test 2.

where conducted. The time passed since the UE is searching for a PLMN until the PDU session establishment is successful is also recorded. RTT measurements are shown in Fig. 5. RTT to the Internet is longer when the connection has to pass through the cloud (slice 1), being affected by the programmed network delay.

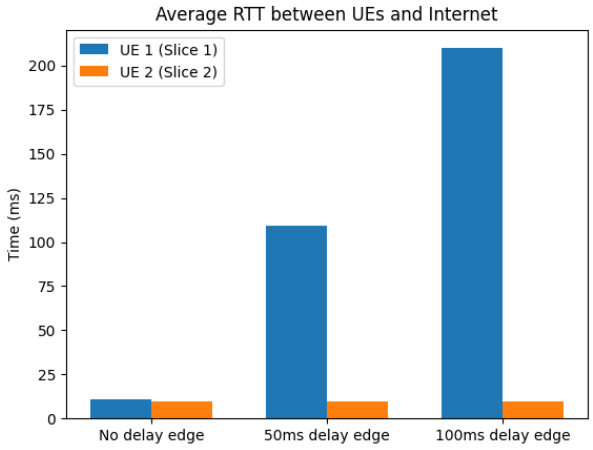

Fig. 5. Average RTT between UEs and Internet

The time the UEs require to establish a connection with the 5G core is shown in Fig. 6. Here we notice the impact of the edge-cloud transmission delay in the control traffic between UEs and Core. There is also an impact in the required time for the slice 2 where the target UPF is at the edge because the control messages between this UPF and the rest of core functions must transit from cloud to edge.

These experiments demonstrate the cloud-native platform capabilities and how it is possible to conduct experiments

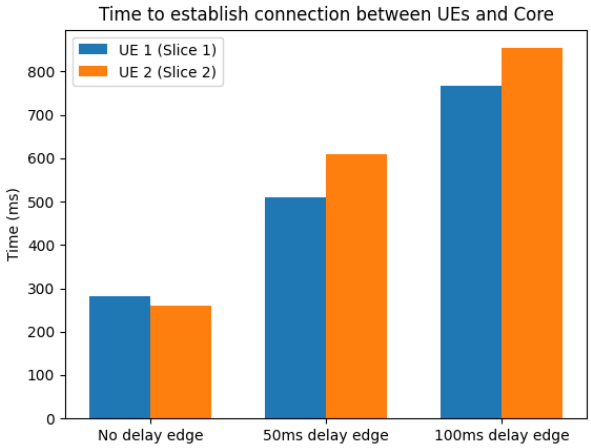

Fig. 6. Connection time UEs

using key elements of the 5G mobile network ecosystem, such as Network Slicing, and how, with simple mechanisms, the capabilities of the network can be affected to simulate different scenarios of interest depending on the experiment.

V. Conclusions and Future Work

With the transition towards cloud-native technologies, there is an increasing interest in the development of 5G infrastructures following this approach. A fully functional 5G Cloud-Native Platform is shown, adapted to today's needs by separating the data plane and the control plane, and offering the ability to investigate different features, such as the use of edge-computing, in a testbed offering experimentation with private 5G networks.

The future work envisages the addition of network virtualisation technologies such as SR-IOV. SR-IOV enables to segment a compliant network device, recognized on the host node as a physical function, into multiple virtual functions, and make them available for direct IO to the Kubernetes Pod. The addition of cloud managed Kubernetes (e.g. AWS EKS) is also on the roadmap. Work will also be done on optimising network resources to provide multiple virtual networks with different bandwidths or QoS dynamically, on optimising containers that require features such as low latency or exclusive access to CPU resources, and on adding new general-purpose hardware to provide a wider catalogue with greater resources for new experiments.

Acknowledgment

This research was supported by the Spanish Centre for the Development of Industrial Technology (CDTI) and the Ministry of Economy, Industry and Competitiveness under grant/project CER-20191015 / Open, Virtualized Technology Demonstrators for Smart Networks (Open-VERSO). We would like also to thank Jorge Arroyo for his help in the different tests carried out to validate our platform.

REFERENCES

[1] S. Van Rossem et al., "A Vision for the Next Generation Platform-as-a-Service," 2018 IEEE 5G World Forum (5GWF), 2018, pp. 14-19, doi: 10.1109/5GWF.2018.8516972.

[2] P. C. Amogh, G. Veeramachaneni, A. K. Rangisetti, B. R. Tamma and A. A. Franklin, "A cloud native solution for dynamic auto scaling of MME in LTE," 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017, pp. 1-7, doi: 10.1109/PIMRC.2017.8292270.

[3] B. Dzogovic, V. T. Do, B. Feng and T. van Do, "Building virtualized 5G networks using open source software," 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2018, pp. 360-366, doi: 10.1109/ISCAIE.2018.8405499.

[4] J. Haavisto, M. Arif et al., "Open-source rans in practice: an over-the-air deployment for 5G MEC," CoRR, vol. abs/1905.03883, 2019.

[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] K. C. Apostolakis et al., "Cloud-Native 5G Infrastructure and Network Applications (NetApps) for Public Protection and Disaster Relief: The 5G-EPICENTRE Project," 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), 2021, pp. 235-240, doi: 10.1109/EuCNC/6GSummit51104.2021.9482425.

[7] K. Haensge, D. Trossen, S. Robitzsch, M. Boniface and S. Phillips, "Cloud-Native 5G Service Delivery Platform," 2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2019, pp. 1-7, doi: 10.1109/NFV-SDN47374.2019.9040042.

[8] S. Vittal, A. Chilukuri, S. Sarkar, A. Shinde and A. Franklin A, "Performance Study of Large Scale Network Slice Deployment in a 5G Core Testbed," 2021 IEEE 4th 5G World Forum (5GWF), 2021, pp. 311-316, doi: 10.1109/5GWF52925.2021.00061.

[9] Multus, A CNI meta-plugin for multi-homed pods in Kubernetes, https://github.com/k8snetworkplumbingwg/multus-cni

[10] Cloud-native Network Function (CNF) Testbed, https://github.com/cncf/cnf-testbed

[11] OpenAirInterface, 5G Wireless Implementation, https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/develop

[12] Container Image Collection of Openverso Project, https://github.com/Gradiant/openverso-images

[13] Helm Charts Collection of Openverso Project, https://github.com/Gradiant/openverso-charts

[14] Ettus Device Plugin for Kubernetes, https://github.com/Gradiant/ettus-device-plugin