Design and Evaluation of an Orchestrated E2E Cloud-Native Mobile Network with O-RAN

Jorge Baranda, Albert Bel, Sergio Barrachina-Muñoz, Miquel Payaró, Josep Mangues-Bafalluy
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Castelldefels, Spain
{jbaranda, abel, sbarrachina, mpayaro, josep.mangues}@cttc.cat

***Abstract*—The architecture and management of next-generation mobile networks are integrating novel networking paradigms to support challenging scenarios. This paper presents the design, implementation, and evaluation of a fully cloud-native, end-to-end mobile network integrating O-RAN functionality. Built on top of open-source software, our proposed framework develops a set of cloud-native deployment artefacts—such as Helm Charts and Open Source MANO packages—also released as open-source. This framework supports highly disaggregated and distributed deployments of key network functions, including the 5G core, gNodeB (with decoupled Centralised and Distributed Unit entities), near-RT RIC, and xApps, enabling agile, portable, and flexible instantiations. Experiments show that the full cloud-native mobile network can be instantiated in less than five minutes, achieving a throughput comparable to *bare-metal* setups. This work highlights significant improvements in deployment efficiency and adaptability, contributing towards automated and autonomous cloud-native Beyond 5G/6G networks.**

***Index Terms*—B5G/6G, O-RAN, E2E Orchestration, Cloud-Native, open-source, Experimental Evaluation, Open5GS, srsRAN, FlexRIC**

I. Introduction

The architecture of next-generation mobile networks, spanning both the Core and the Radio Access Network (RAN) segments, is experiencing a deep transformation driven by the adoption of cloud-native principles and open interfaces, as investigated in the UNICO-5G I+D 6G-BLUR project [1]. These advancements enable the disaggregation of monolithic components into distinct softwarised network functions (NFs), the separation of control and user planes, and promotes interoperability among multi-vendor providers. This transformation offers increased network management capabilities in terms of flexibility and dynamicity, enabling customized deployments for emerging use cases and scenarios. These include those proposed by vertical industries and support the development of new business models, such as Non-Public Networks.

Focusing on the deployment process, this paper presents the design and development of a fully cloud-native, end-to-end (E2E) mobile network incorporating Open RAN (O-RAN) functionality. The proposed approach builds on open-source software projects to support the disaggregated, distributed, and sequential deployment of mobile network entities, relying on appropriate management and orchestration (MANO) procedures to enable automation and portability.

The main contributions of this paper are threefold. First, it provides a description of the process of creating highly configurable cloud-native artefacts to be used with a custom MANO system to automate the deployment of a fully disaggregated mobile network (i.e., from the core to the RAN) compatible with commercial Radio Units (RU) employing Split 7.2 across distributed Points of Presence (PoPs). Second, these artefacts and the MANO system are released as open-source to support reproducibility and extension. Third, the paper evaluates deployment times for each mobile network entity, deriving the overall E2E deployment time, and compares throughput performance between the proposed cloud-native and *bare-metal* approaches using the same hardware equipment.

Experimental results highlight the efficiency of the proposed methodology, showcasing that a complete cloud-native mobile network—including the mobile core, gNodeB (gNB) both monolithic and decoupled into Centralised Unit (CU) and Distributed Unit (DU) entities, near-Real Time (near-RT) Radio Intelligent Controller (RIC), and monitoring xApps—can be instantiated from scratch in less than five minutes employing an ETSI NFV-based MANO platform. Notably, the performance of this cloud-native deployment in terms of achieved throughput is comparable to that of *bare-metal* deployments. Furthermore, the proposed approach combining our custom MANO platform and the generated artefacts allow dynamic deployments, such as the on-demand creation and deletion of different instances of entities like UPFs, gNBs and xApps, which exceeds current capabilities of similar approaches in the literature.

The remainder of this paper is structured as follows. Section II reviews related work. Section III provides details about the methodology used to transition from *bare-metal* to cloud-native deployments. Section IV presents an experimental evaluation assessing our cloud-native approach. Finally, Section V concludes the paper.

II. Related work

During last years, 5G research has been boosted by the development of testbeds, offering a controlled environment to evaluate the capabilities and limitations of different 5G technologies and architectures. To build such testbeds, researchers mostly rely on open-source software, due to reduced costs and innovation opportunities through customization. Indeed, there are several open-source projects to build the Core (e.g.,

This work has received funding from Spanish MINECO grants TSI-063000-2021-56/TSI-063000-2021-57 (6G-BLUR SMART/6G-BLUR JOINT), UNITY-6G project, funded by European Union's HE SNS JU research and innovation programme under the Grant Agreement No 101192650, Grant PID2021-126431OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and Generalitat de Catalunya grant 2021 SGR 00770.

Open5GS, Open Air Interface (OAI), Free5GC) and RAN segments (e.g., srsRAN, OAI, UERANSIM) of a mobile network. Among these options, our work considers Open5GS and srsRAN as suitable choices due to their relative ease of deployment and performance, as identified in [3]. Most of these testbeds focus on the performance evaluation of 5G network deployments in terms of metrics like throughput or latency [3]–[5] and are built running these open-source projects as *bare-metal* processes. However, as previously mentioned, the evolution in the architecture of next generation mobile networks advocates for the exploitation of cloud-native principles for the deployment of such environments, which is the focus of this work. Such approach provides faster, continuous software update, high availability, resilience, flexible, distributed and portable deployments easier to replicate.

Recent literature presents several examples on the deployment of a cloud-native 5G core using open-source software [6]–[9]. In [6], the 5G core network is deployed as a single container in a Kubernetes cluster using OAI software. In [7], [8], the 5G core network, based on Open5GS [13] software, is deployed as multiple containers at multiple nodes of the same Kubernetes cluster. However, the distribution of the different NFs of the mobile core is predefined and static for a single Kubernetes cluster, requiring adaptations for each deployment environment. The same approach, a manual deployment with Docker of a Free5GC based mobile core, is considered in [9]. Although [7], [9] deal with core network deployments considering multiple User-Plane Functions (UPFs) instances, which is not the case of [8], they are deployed all-at-once, lacking the flexibility our work provides to further redefine the deployment by starting them upon demand. Regarding the RAN segment, cloud-native implementations are limited. Relevant examples are [2], [10], which deploy gNBs relying on UERANSIM with emulated transmission and OAI using Split 8 for over-the-air (OTA) communication, respectively. However, there is no existing work demonstrating a cloud-native RAN deployment using srsRAN project. Moreover, other O-RAN entities, such as the near-RT RIC or the xApp, are executed as *bare-metal* processes [11], [12].

Our work provides a step forward by introducing a methodology to build a fully E2E cloud-native mobile network with enhanced automation MANO capabilities based on open-source projects. To the best of our knowledge, it is the first work to enable disaggregated and sequential deployments—from Core to RAN—, while integrating O-RAN components across distributed independent Kubernetes clusters and supporting Split 7.2 with commercial radio units (RUs). We also provide an assessment of deployment times, which is lacking in current literature, and compare the obtained throughput performance against a *bare-metal* setup, demonstrating the suitability of our proposed approach to build not only experimental testbeds but also small-scale 5G private networks.

III. System Design

This section, first, outlines the methodology adopted to transition from a manual *bare-metal* deployment to an automated, and fully disaggregated cloud-native approach, employing MANO tools aligned with ETSI NFV specifications, such as Open Source MANO (OSM). Second, we present the characteristics of the generated key artefacts, i.e., Helm Charts, based on the characteristics of the selected open-source solutions: Open5GS [13] for the core, srsRAN [14] for the RAN, and Flexric [15] for O-RAN functionalities.

A. Methodology

The transition from a *bare-metal* setup to an automated, cloud-native deployment of an E2E mobile network involves several key steps. First, we reviewed available documentation to identify which mobile NFs require external access to allow a distributed environment and the required specific network ports to expose. Next, a test environment was configured to run the open-source *bare-metal* processes and analyse its configuration capabilities in a distributed setup, moving beyond a simple *all-in-one* deployment. This enabled the identification of challenges associated with containerizing the software under orchestration. In this phase, we analyse the control-plane signalling between mobile NFs -such as the AMF and gNB/CU— using tools like Wireshark, particularly during deployment and user equipment (UE) attachment. Our analysis revealed that the open-source processes typically bind to the host network interface, which works in bare-metal environments but presents challenges in cloud-native contexts. Binding to container interfaces can complicate inter-node communication due to private cluster addressing, potentially requiring complex routing configurations and posing security risks by exposing internal workloads.

The third step involves creating Docker containers and publishing them in a repository. These containers include not only the needed open-source code to run each mobile NF entity, but also code patches possibly developed derived from the previous step and custom scripts. These scripts dynamically adjust configuration parameters based on values assigned to the container when launched by the container orchestrator. The next step is the generation of mobile NF artefacts, such as Helm Charts and OSM packages. Helm Chart templates are embedded within the virtual NF packages included in the OSM Network Service (NS) package. In this phase, appropriate parametrization of the Helm Chart configuration settings is essential to enable flexible, customized, and automated deployment using OSM. Finally, we developed an orchestrator framework on top of OSM, depicted in Fig. 1, to simplify the deployment process and to help users without deep technical expertise to create their own E2E mobile network instances in a distributed environment. This orchestrator presents a Graphical User Interface (GUI) that enables the user to perform the on-demand (i.e., in different time instances) instantiation and termination of multiple instances of the NFs (e.g., UPFs, gNBs, xApps) as NSs across distributed PoPs.

B. Implementation: Artefact development

This subsection describes the characteristics of the developed blueprints (i.e., Helm Charts) for the different mobile NFs based on the selected open-source projects. These Helm

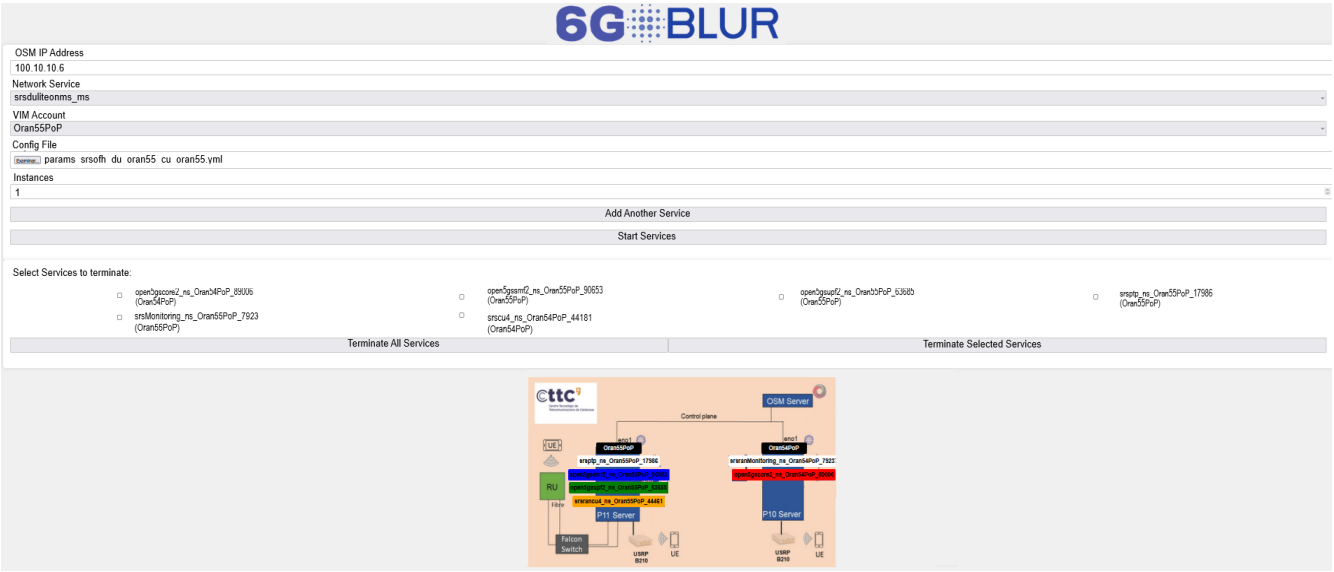

Fig. 1. 6G-BLUR E2E Mobile Network Orchestrator graphical user interface.

charts, along with the mentioned orchestrator framework, are publicly available in [16].

1) 5G Mobile Core: The decoupling of control-plane (CP) and user-planes (UP) of the 5G mobile core is achieved by means of three configuration blueprints. These Helms charts use the same Docker container image running *Open5GS v2.7.1* software. The first blueprint defines the Kubernetes templates for CP functions (e.g., AMF, AUSF, BSF, NRF, PCF, SCP, NSSF, etc.), each one running as an independent Kubernetes pod executing the corresponding NF process. This blueprint serves as a centralized point of management for the mobile core in a central cloud. The second blueprint contains the templates to deploy the Session Management Function (SMF), which has been disaggregated from the CP blueprint to allow a flexible activation/deactivation of slice components [20]. This blueprint is complemented by the third blueprint, handling the deployment of the UP by means of the User-Plane Function (UPF), thus providing flexibility by enabling the dynamic activation or deactivation of UPF instances at cloud, regional or edge PoPs as needed according to the requirements of the scenario [17]. The deployment of SMF instances depend on the slice configuration set at the AMF entity of the CP blueprint, and accordingly, the deployment of UPF instances depend on the configuration set at the SMF component/s. The designed blueprint accept a dynamic number of such relevant parameters so later, this can be dynamically customised through OSM instantiation commands. These blueprints provide external access for the pods by exposing standard ports defined in 3GPP specifications and making use of Kubernetes *LoadBalancer* type of service. Among the most relevant ports, we find those related to the interconnection between RAN and the Core, that is 38412 for SCTP (N2 interface), 2152 for GTP (N3, N4 interfaces) and 8805 for UDP connection (N4 interface).

2) O-RAN Disaggregation: For the RAN segment, three different blueprints have been developed to allow multiple deployment schemes using *srsRAN* software: i) a compact gNB, or ii) a pair of independent DU and CU entities. Moreover, in the case of gNB and DU, multiple versions of the artefacts are generated to support multiple *"radio"* configurations. These configurations include the use of an emulated RU using Zero-Messaging Queue (ZMQ) library, an SDR-based RU (e.g., USRP B210 for Split 8), or an O-RU using the Open Fronthaul (O-FH) interface Split 7.2 interacting with commercial RU, which is a feature not found in the literature. Using the FlexRIC framework (*br-flexric branch*), additional blueprints were designed for the deployment of the near-RT RIC and monitoring xApps [19]. The near-RT RIC blueprint allows the interaction with the gNB/CU/DU entities via the O-RAN E2 interface over SCTP protocol using port 36421. The monitoring xApp blueprint retrieves Key Performance Measurement (KPM) data from these entities using a SCTP connection over port 36422. It is worth mentioning that for the case of O-RU deployments employing Split 7.2, an additional Helm Chart has been generated to acquire the tight timing synchronization by means of the Precision Time Protocol (PTP) between the DU and the RU. This synchronization is provided using an O-RAN capable switch interconnecting the O-RU with the host where cloud-native gNB or DU instances are deployed. The CU blueprint enables northbound connectivity to the mobile core via N2 and N3 interfaces, and southbound communications with the DU via F1-c (SCTP, port 38472) and F1-u (GTP, port 2152) interfaces. To prevent port conflicts in the cloud-native deployment of CU in distributed setups, where the CU connects to both the DU and the UPF via port 2152, a patch was applied to the srsRAN software, allowing the use of separate GTP ports for the F1-U and N3 interfaces. Additionally, the DU blueprint supports flexible configuration parametrization to allow multiple DU instances to connect to a single CU instance, in alignment with the deployment scenarios defined in the O-RAN specifications [18].

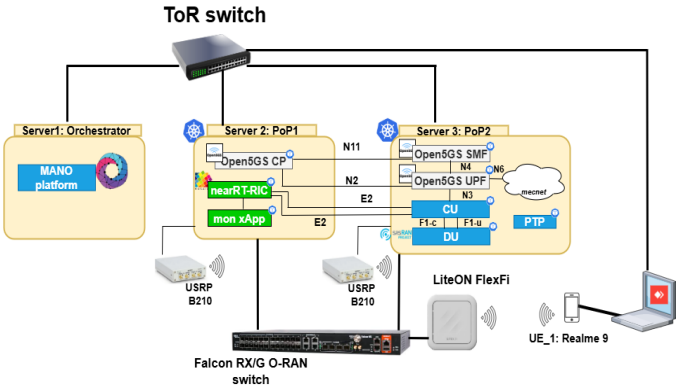

Fig. 2. Experimental setup for the distributed orchestration of end-to-end mobile networks.

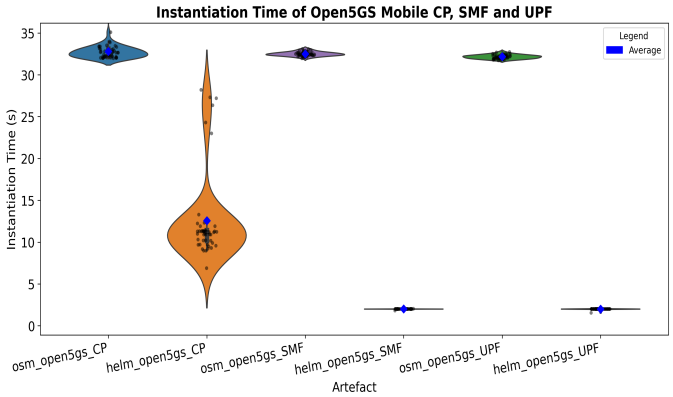

Fig. 3. Instantiation time distribution of Open5GS CP, SMF and UPF artefacts.

IV. Experimental Evaluation

This section introduces the experimental testbed and evaluates the cloud-native artefact instantiation time and the throughput performance of the proposed cloud-native deployment versus the execution of the same software running as *bare-metal* processes in the same hardware setup.

A. Testbed Setup and Experiment Description

Fig. 2 shows the experimental setup deployed in xG EXTREME Testbed located in CTTC premises. It consists of three commercial of-the-shelf servers (COTS), two USRPs B210, a Falcon RX/G O-RAN switch and PTP grandmaster, a LiteON FlexFi hardware acting as O-RU Split 7.2, and a Realme 9 smartphone connected to a laptop so it can be interfaced via a remote desktop tool (e.g., AnyDesk). The three COTS are a 24 Intel CPU@3.00 GHz, 64GB RAM, 500 GB HDD server running Ubuntu 22.04 operative system. Among them, one executes the MANO platform depicted in Fig. 1, powered by an instance of OSM Release 14. The remaining two servers are the PoPs, where a single-node Kubernetes cluster in each server acts as the Container Infrastructure Manager. Fig. 2 also includes the mobile network entities' distribution followed in this experimentation. Notice that this is an arbitrary distribution and it is fully configurable through our MANO system, thus allowing complex scenarios including additional Kubernetes clusters acting as PoPs as those described in O-RAN specifications [18]. In order to give statistical meaning to the results, each instantiation experiment is repeated 50 times, and the results are presented using a violin plot depicting the distribution of the obtained results.

B. Instantiation time assessment

Fig. 3 presents the time required to instantiate the Open5GS CP, SMF and UP artefacts. It compares an instantiation using our MANO system relying on OSM application programming interface (API) to control our set of distributed PoPs from a central point with respect to using Helm command line interface (CLI) executed directly in a given Kubernetes cluster.

From Fig. 3, it can be observed that the mean time required for OSM to declare as instantiated the Open5GS CP, the SMF and UPF NSs is on average around 32 seconds, while for the Helm CLI is around 12 seconds for the CP and 3 seconds for the SMF and UPF Helm Charts, respectively. Comparing instantiation methods, the required time to instantiate the CP, the SMF and UPF artefacts using OSM is quite stable through repetitions, as seen from the shape of the violin plot. Although the underlying Helm Chart used in the OSM package is the same as the one used with Helm CLI, the difference in experienced time between using OSM or using the Helm CLI comes due to how OSM handles the interactions with Kubernetes. The OSM plugin for Kubernetes goes beyond checking container deployment status and performs additional actions (e.g., readiness for Day-1 and Day-2 options). However, the generated script to measure the time with the deployment based on Helm CLI just measures the time it takes for Kubernetes to declare all the pods associated to the Helm Chart in *Ready* state. Furthermore, the difference between CP, SMF and UPF instantiation times comes from the fact that when starting the Helm Chart associated to the CP, up to 10 different Kubernetes pods, services and other Kubernetes resources are started parallelly, one for each CP NF. In the case of the SMF and UPF using Helm CLI, only a single Kubernetes pod and service resources are started. In addition to this, some of the pods in the CP Helm Chart require to re-start multiple times due to dependencies between mobile core NFs, such as, for instance, the need to contact and register into the Network Repository Function (NRF). This would also explain the higher experienced variability (i.e., shape of the violin plot) for the instantiation of the Open5GS CP using Helm CLI, where the maximum experienced instantiation is closer to the average one experienced with OSM. A final remark comes when comparing the Kubernetes-based results with the instantiation of a similar mobile core NS using virtual machines (VMs). As reported in [21], the time required to deploy a 4G mobile core NS using a single VM containing the multiple NFs (CP and UP) jumps up to 150 seconds. In the work herein analysed, the combined time with our MANO platform (CP, SMF and UPF) would be in the order of 96 seconds (i.e., 32 x 3 seconds), around 40% of reduction, without mentioning that with the produced artefacts a more flexible, distributed, and sequential deployment can be achieved.

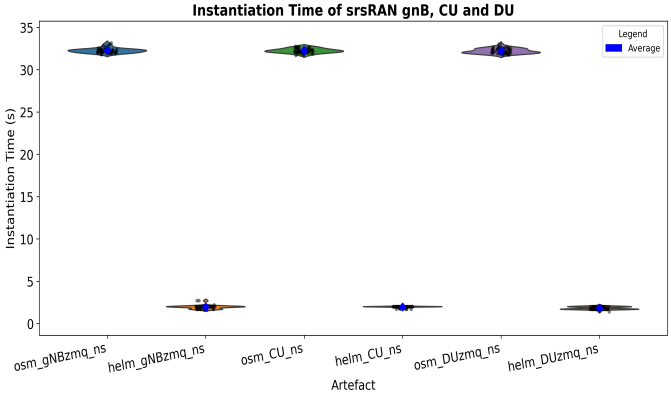

Fig. 4. Instantiation time distribution of srsRAN-based gNB, CU and DU artefacts.

Fig. 4 and Fig. 5 present the same analysis for the O-RAN entities, considering a compact gNB, the decoupled CU/DU entities, the nearRT-RIC and a monitoring xApp. To conduct these experiments, i) we used the ZMQ-version of gNB and DU artefacts to avoid stressing the radio hardware, (i.e., USRP B210 or LiteON FlexFi), although parallel isolated experiments using these hardware components produced similar results; ii) the previously developed core CP, SMF and UPF artefacts were deployed to provide a functional mobile core for attaching the RAN components; and iii) a cloud-native instance of the near-RT RIC is also available when running the experiments with the gNB, CU, DU and xApp to validate the proper establishment of E2 protocol handshakes.

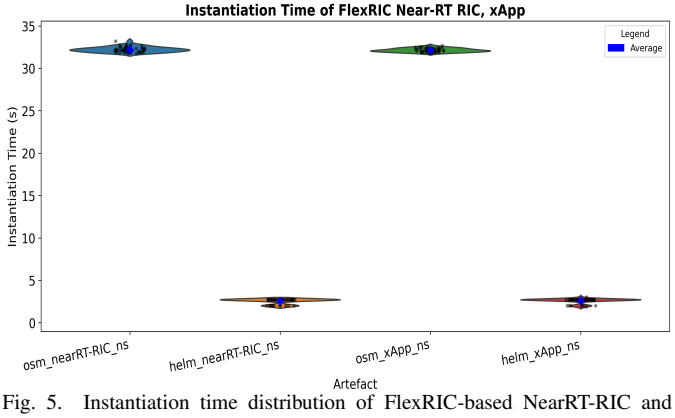

Fig. 5. Instantiation time distribution of FlexRIC-based NearRT-RIC and xAPP artefacts.

The observed trends align with those previously reported in Fig. 3. Specifically, the average instantiation times using OSM is approximately 32 seconds, while the time required by the Helm install CLI operation to declare the associated pods in *Ready* state is around 3 seconds. Bear in mind that in all these artefacts, there is only a single pod, and a single service Kubernetes resource defined, like in the Open5GS SMF and UPF artefacts. This simpler Helm Chart definition contributes to lower variability in the measured instantiation times, according to the shape of the violin plot. A final observation is that, despite the size of the container image for srsRAN-based artefacts is nearly three times bigger than the one used in Open5GS artefacts (1.45GB vs 566MB), no noticeable impact on instantiation time is observed. This is attributed to the fact that container images are locally cached within Kubernetes cluster, thus eliminating image pull delays during instantiation repetitions.

Considering all the obtained results, we conclude that a complete functional cloud-native E2E mobile network including O-RAN functionality can be distributively deployed in less than five minutes. Moreover, the modularity of the created artefacts and the capabilities of our MANO system facilitate on-demand deployments, that is, new instances of gNBs, DUs, xApp can be activated (or terminated) in less than one minute thus modifying the mobile network topology as needed.

C. Performance Comparison: cloud-native vs bare-metal

In this section, we compare the experienced throughput of our orchestrated E2E cloud-native deployment with that of an equivalent manual setup running the same Open5GS and srsRAN components as *bare-metal* processes on the PoP servers, following the same distribution illustrated in Fig. 2. These measurements are done considering the CU and DU running as two independent processes, using the LiteOn FlexFi RU (i.e., Split 7.2 between DU and RU) and about one-meter distance between the RU and the smartphone with OTA transmission, thus experiencing good propagation conditions. The DU is configured in Time Division Duplexing (TDD) mode, using a transmission bandwidth of 20MHz, a 30KHz subcarrier spacing with a Time Division Duplexing (TDD) pattern DDDDDDSUUU, using two antenna ports for the downlink (DL) and one antenna port for the uplink (UL) and a QAM64 modulation and coding scheme (MCS) table both for the physical downlink/uplink shared channels (PDSCH/PUSCH). The application used to measure the throughput is Iperf3.

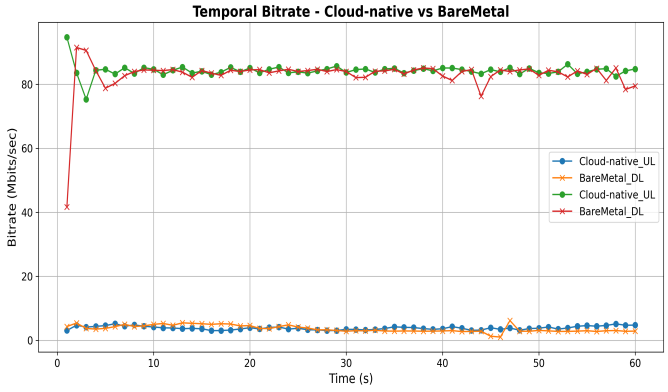

Fig. 6. Bit rate temporal evolution in UL and DL: cloud-native vs *bare-metal*.

Fig. 6 presents the throughput evolution during a sixty seconds OTA Iperf session using TCP in both DL and UL directions. The DL case (i.e., server: smartphone, client: UPF) and the UL case (i.e., server: UPF, client: smartphone) illustrate a clear performance gap —approximately 80 Mbps in DL versus 4 Mbps in UL—primarily due to the TDD slot allocation and the asymmetry in antenna port configurations. The most interesting aspect is that for this Iperf session, the obtained throughput performance using the cloud-native

approach is comparable to that of the *bare-metal* execution. These results indicate that the cloud-native approach seems not to experience a penalty in terms of performance. This can be attributed to the fact that generated Helm Chart artefacts are not introducing any constraint in terms of use of CPU and RAM resources from the host system.

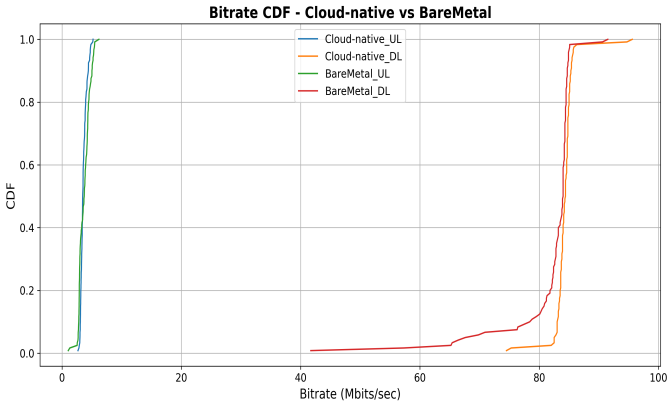

Fig. 7. Bit rate CDF in UL and DL: cloud-native vs *bare-metal*.

To verify this trend, we repeated ten times the previous experiment. Fig. 7 presents the obtained cumulative distribution function (CDF) of the experienced throughput. As observed, the cloud-native approach obtains an equivalent performance to the *bare-metal* execution. Indeed, for the DL case, the *bare-metal* approach presents more variability. This can be explained due to the impact of adaptive coding and modulation algorithm in the different repetitions, which increases and decreases the MCS depending on the channel conditions experienced during experiment execution. Overall, the cloud-native approach can be considered a viable option for deploying 5G mobile network components, especially when balanced against the operational flexibility, scalability and portability capabilities it provides.

V. Summary and Conclusions

This paper presents the design, implementation, and evaluation of a fully cloud-native E2E mobile network built using open-source software components. A core contribution is the proposed methodology for generating a comprehensive set of deployment artefacts— including Helm Charts and OSM packages—that enable agile, distributed, and customizable E2E deployments. These artefacts, released as open-source alongside with our MANO system, facilitate the seamless and portable on-demand deployment of disaggregated and distributed mobile networks, suitable for experimental testbeds and small-scale 5G private networks.

Experimental results show that the entire cloud-native mobile network can be instantiated from scratch in under five minutes. This includes the instantiation of the mobile core (CP and UP), the CU, the DU, the near-RT RIC, and a monitoring xApp using our MANO system relying on OSM. Moreover, the cloud-native deployment achieves a similar throughput performance when compared to a *bare-metal* setup, demonstrating its practicality and efficiency.

The proposed methodology effectively employs the evolving capabilities of mobile network open-source projects, empowering B5G/6G network management to dynamically adapt and scale based on operational requirements. As future work, we plan to enhance the proposed MANO system by integrating closed-loop feedback mechanisms to enable autonomous orchestration decisions, thus approaching the flexibility and efficiency required in modern mobile networks.

References

[1] UNICO-5G I+D 6G-BLUR Project: Joint RAN and transport network control/orchestration mechanisms, Available at: https://6g-blur.cttc.es/ [Accessed: June 2025]

[2] V. Q. Pham, A. Kak, H. T. Thieu, N. Choi, "ORCA: Cloud-native Orchestration and Automation of E2E Cellular Network Functions and Slices", in IEEE WKSHPS Infocom'2024, Vancouver, Canada.

[3] J. E. Håkegård, H. Lundkvist, A. Rauniyar and P. Morris, "Performance Evaluation of an Open Source Implementation of a 5G Standalone Platform", in IEEE Access, vol. 12, pp. 25809-25819, 2024.

[4] D. Pineda, R. Harrilal-Parchment, K. Akkaya, A. Ibrahim, A. Perez-Pons, "Design and Analysis of an Open-Source SDN-based 5G Standalone Testbed", in IEEE INFOCOM WKSHPS, Hoboken, USA, 2023.

[5] L. Phan, D. Pesch, U. Roedic, C. J. Sreenan, "Building a 5G Core Network Testbed: Open-Source Solutions, Lessons Learned and Research Directions", in IEEE ICOIN'24, Ho Chi Minh City, Vietnam, 2024.

[6] O. Arouk and N. Nikaein, "5G Cloud-Native: Network Management & Automation", in IEEE/IFIP Net. Operations and Man. Symposium (NOMS), Budapest, Hungary, April 20-24, 2020. IEEE, 2020, pp. 1–2.

[7] S. Barrachina-Muñoz, M. Payaró, and J. Mangues-Bafalluy, "Cloud-native 5G experimental platform with over-the-air transmissions and end-to-end monitoring", in IEEE CSNDSP'22, Porto, Portugal, 2022.

[8] N. Apostolakis, M. Gramaglia, and P. Serrano, "Design and Validation of an Open Source Cloud Native Mobile Network", in IEEE Communications Magazine, vol. 60, no. 11, pp. 66–72, 2022.

[9] J. Wu, Y. Gao, L. Wang, J. Zhang, and D. O. Wu, "How to Allocate Resources in Cloud Native Networks Towards 6G", in IEEE Network, vol 38., no. 2, pp. 240-246, March 2024.

[10] M. Santos de Brito, E. Modroiu, B. Q. Le, T. Magedanz, M. Corici, J. Mwangama, "Open6GNET: A Learning and Experimentation Platform Based on Open-Source Solutions and Cloud-Native principles", in IEEE INFOCOM WKSHPS, Vancouver, Canada, 2024.

[11] N. Godinho et al., "OREOS: Demonstrating E2E Orchestration in 5G Networks with Open-Source Components," in IEEE INFOCOM WKSHPS, Hoboken, USA, 2023, pp. 1-2.

[12] A. Tripathi et al., "End-to-End O-RAN Control-Loop For Radio Resource Allocation in SDR-Based 5G Network", IEEE Military Comms. Conf. (MILCOM'23), Boston, MA, USA, 2023, pp. 253-254.

[13] Open5GS 5G Core, Available at: https://open5gs.org/open5gs/docs/guide/01-quickstart/ [Accessed: June 2025].

[14] srsRAN Project - Open Source RAN, Available at: https://github.com/srsran [Accessed: June 2025].

[15] FlexRIC, O-RAN Alliance compliant nearRT-RIC, xApps, Available at: https://gitlab.eurecom.fr/mosaic5g/flexric [Accessed: June 2025].

[16] 6GBLUR Gitlab repository, Available online at: https://gitlab.cttc.es/mdi-6gblur/joint [Accessed: June 2025]

[17] J. Baranda, S. Barrachina, R. Nikbakht, M. Payaró, J. Mangues-Bafalluy, "Disaggregating a 5G Non-Public Network via On-demand Cloud-Native UPF Deployments", in IEEE CNSM'23, Niagara Falls, Canada.

[18] O-RAN Alliance, "O-RAN Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN 6.0," 2024. [Online]. Available at: https://specifications.o-ran.org/specifications.

[19] J. Baranda, A. Bel, S. Barrachina-Muñoz, M. Payaró, J. Mangues-Bafalluy, "Distributed Sequential Cloud-Native Deployment of an End-to-End 5G Network with O-RAN Functions", in IEEE NoF24, Castelldefels, Spain.

[20] J. Baranda, A. Bel, S. Barrachina-Muñoz, M. Payaró, J. Mangues-Bafalluy, "End-to-End Slice Orchestration in a 5G cloud-native Mobile Network with O-RAN Split 7.2", submitted to IEEE INFOCOM'25, London, UK, 2025.

[21] J. Baranda, J. Mangues, R. Martínez, L. Vettori, K. Antevski, C.J. Bernardos, X. Li, "Realizing the Network Service Federation Vision: Enabling Automated Multidomain Orchestration of Network Ser-vices", in IEEE Vehicular Tech. Mag., vol. 15, no. 2, pp. 48-57, 2020.