

Experimental 5G RAN Slicing deployment with SDRs in a Cloud-Native Testbed

Mariana Ribeiro, Lucas Costa, Cleverson Nahum, Kauan Miranda, Miguel Lunardi, Cristiano Both, Silvia Lins, Andrey Silva, Aldebaro Klautau

Abstract—Integrating latency-sensitive Software-Defined Radio (SDR) into cloud-native Fifth Generation (5G) environments for Radio Access Network (RAN) Slicing represents a great challenge, often overlooked by existing testbeds. This paper proposes a reproducible, dual-layered virtualized architecture addressing this gap. We combine hypervisor-level virtualization (Proxmox Virtual Environment (VE) with Peripheral Component Interconnect Express (PCIe) passthrough) for deterministic Universal Software Radio Peripheral (USRP) B210 performance with Operating System (OS)-level container orchestration (Kubernetes and Helm). Customizing OpenAirInterface (OAI) and ORANSlice, our system enables autonomous, policy-driven slice management. Experimental evaluations with commercial 5G User Equipment (UEs) demonstrate the architecture’s ability to dynamically reallocate throughput in real-time while maintaining strict computational efficiency and stability without external controllers.

Keywords—5G Networks, Cloud-Native Architecture, RAN Slicing, Testbed, Virtualization.

I. INTRODUCTION

The evolution of Fifth Generation (5G) and emerging Sixth Generation (6G) mobile networks is fundamentally driven by the need to support highly diverse use cases, ranging from massive Internet of Things (IoT) deployments to ultra-reliable low-latency communications. A critical enabler of this flexibility is Radio Access Network (RAN) slicing, which enables the instantiation of multiple logically isolated networks on a shared physical infrastructure [1]. This paradigm enables granular control over resource allocation and service treatment. For instance, a slice dedicated to real-time communication can be dynamically allocated the exact resources required to satisfy its strict low-latency demands [1].

The deployment and management of RAN slicing scenarios in research environments rely on virtualized and orchestrated infrastructures. In this context, the broader transition towards cloud-native architectures has revolutionized network management, leveraging software-centric tools like Kubernetes and Helm charts facilitates the automated deployment, granular monitoring, and rapid scalability of complex network functions. To evaluate these orchestrated topologies, experimental

testbeds frequently utilize Radio Frequency (RF) simulators, such as the OAI-RF within the OpenAirInterface (OAI) framework. While these tools offer a highly controlled baseline, they inherently abstract real-world hardware constraints and physical channel impairments. Consequently, integrating physical Software-Defined Radios (SDRs) into the testbed significantly enhances test fidelity, providing the empirical data necessary to validate the real-world applicability of the proposed slicing studies.

A major technical challenge remains reconciling the practical fidelity of physical SDR devices with the scalability, flexibility, and portability offered by cloudified architectures. Current experimental platforms for 5G RAN slicing are predominantly designed for bare-metal servers [2], [3], lacking clear solutions for integrating latency-sensitive RAN equipment into fully virtualized environments [4], [5]. Furthermore, they neglect guidelines for establishing dual-layered virtualization setups in which hardware traverses both hypervisor Virtual Machines (VMs) and containers at Operating System (OS) level without violating strict deterministic timing constraints.

To address this challenge, we propose a functional, highly reproducible 5G testbed that integrates real SDR hardware into a cloud-native ecosystem. By detailing hypervisor-level hardware passthrough techniques alongside Kubernetes and Helm orchestration, our solution demonstrates how a dual-layered architecture can harness both the practical realism of real SDR equipment and the operational agility of a virtualized setup, ultimately allowing modular control of containerized network functions and their underlying VMs.

II. PROPOSED METHOD

We propose the design and deployment of an experimental 5G network that leverages virtualization and container orchestration, alongside physical Universal Software Radio Peripheral (USRP) hardware, to enable and evaluate RAN slicing. The underlying infrastructure is built upon the Proxmox Virtual Environment (VE), which provides the hypervisor layer for provisioning and managing the VMs that hosts the network topology. To orchestrate the diverse network functions, we deployed a Kubernetes cluster across these VMs. This containerized approach facilitates the scalable deployment and management of both the 5G Core and the RAN, utilizing the OAI open-source platform [6].

We divided the Kubernetes cluster across two primary VMs, as illustrated in Figure 1. The first VM is dedicated solely to running the Next-Generation NodeB (gNB) pod, since the

Mariana Ribeiro, Lucas Costa, Cleverson Nahum, Kauan Miranda and Aldebaro Klautau are with Universidade Federal do Pará (UFPA), Belém, Brazil. Silvia Lins and Andrey Silva are with Ericsson Research, Brazil. Miguel Lunardi and Cristiano Both are with University of Vale do Rio dos Sinos (UNISINOS), São Leopoldo, Brazil. Emails: [mariana.ribeiro@itec.ufpa.br, lucas.queiroz.costa@itec.ufpa.br, cleversonnahum@ufpa.br, kauan.tavares@itec.ufpa.br, aldebaro@ufpa.br, silvia.lins@ericsson.com, andrey.silva@ericsson.com, lunardimiguel@edu.unisinos.br, cbboth@unisinos.br].

system integrates a real SDR for RAN operations, and this hardware is exclusively mapped to that VM. The second VM operates as both a control plane and a worker node, dedicated exclusively to 5G Core network functions. This topological separation eliminates resource contention, ensuring the gNB maintains uninterrupted access to its required computational resources.

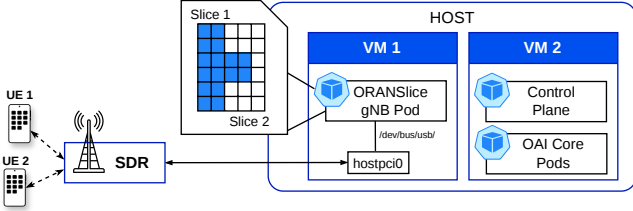


Fig. 1. Architecture of the proposed dual-layered virtualized 5G testbed. Proxmox VE manages the hypervisor-level VMs, using PCIe passthrough for deterministic SDR performance. Concurrently, Kubernetes orchestrates the containerized OAI 5G Core and slice-enabled gNB.

To introduce slicing capabilities, we integrated the ORANSlice framework [7], which supplies the necessary source code to extend the standard OAI protocol stacks. However, its default distribution lacks native support for the virtualization and Kubernetes-based orchestration of a slice-enabled gNB. To bridge this challenge and fully integrate the framework into our orchestrated setup, we designed a custom Dockerfile to compile and containerize the specialized gNB. After storing this custom image in a local Kube-registry, the baseline OAI Helm charts were significantly modified to deploy this tailored container in place of the default monolithic gNB. In this native configuration, the slicing policy is governed by a configuration file within the gNB pod. The Medium Access Control (MAC) layer periodically reads this file, allowing for the dynamic adjustment of slice throughput allocation via parameters such as `rRMPolicyMinRatio` and `rRMPolicyMaxRatio`.

The integration of the USRP into the virtualized environment requires a high-performance interface to satisfy the stringent real-time demands of 5G signal processing. Within the Proxmox VE framework, we evaluated two primary methods for hardware mapping: standard Universal Serial Bus (USB) device passthrough and Peripheral Component Interconnect Express (PCIe) passthrough via Input-Output Memory Management Unit (IOMMU) technology. While both methods can expose a USB device to a guest, they differ fundamentally in whether the hypervisor emulates the USB controller (standard passthrough) or assigns the physical controller directly to the VM (PCIe passthrough).

The standard USB passthrough maps the device by its ID or physical port. Despite its versatility and support for hot-plugging, this method leaves the physical USB port under the host's control, requiring the hypervisor to translate every transfer between the guest's emulated controller and the underlying hardware. Although the guest OS may report a USB 3.0 connection, the additional software translation layer introduces variable latency and prevents the deterministic, low-latency bulk transfers required by the USRP. In

initial OAI gNB deployments, this overhead caused critical `ERROR_CODE_OVERFLOW` failures due to the system's inability to process radio samples in real time.

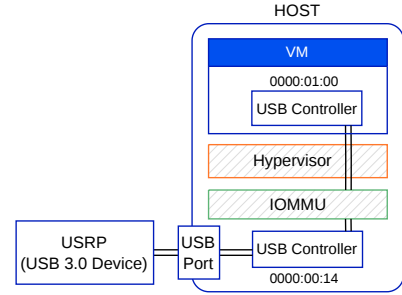


Fig. 2. PCIe passthrough mapping via IOMMU. By bypassing the hypervisor's emulated USB controller, the guest VM gains direct, exclusive access to the physical USRP B210. This direct mapping eliminates software translation overhead, ensuring the strict determinism required to prevent synchronization errors during real-time baseband processing.

To overcome these timing constraints, we implemented the PCIe passthrough approach. As illustrated in Figure 2, this technique leverages IOMMU to isolate the memory address of the USB controller, granting the guest VM direct and exclusive access to the physical hardware. By bypassing the hypervisor's emulated USB stack entirely, the guest VM communicates directly with the physical USB controller via native drivers, achieving near bare-metal performance. Although it strictly binds the hardware to a single VM, this method proved essential for eliminating synchronization errors and ensuring the stability of the experimental 5G RAN.

We decoupled the configuration file from the container image and injected it at runtime via Kubernetes ConfigMaps to efficiently manage the gNB parameters. The default ORANSlice settings were tailored for our testbed: RF parameters were adapted for the USRP B210 (replacing the default X310), and hardcoded Internet Protocol (IP) addresses for the Access and Mobility Management Function (AMF) and N2/N3 interfaces were replaced with environment variables to enable cross-domain connectivity. Moreover, a persistent challenge in modern 5G testbeds is the lack of reproducible methodologies for integrating physical RAN equipment into fully virtualized ecosystems. Most existing experimental platforms either bypass virtualization entirely for their SDRs or present high-level architectures that fail to specify how physical radios are managed within the virtual domain. By detailing the convergence of PCIe passthrough and OS-level container orchestration, we address this oversight in the literature and offer a structured approach to a hardware integration challenge that is frequently overlooked in current experimental platforms.

III. EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

The Proxmox VE testbed infrastructure is distributed across two specialized VMs. The gNB and physical SDR operate on the first VM with 6 Virtual Central Processing Units (vCPUs), 16 GB Random-Access Memory (RAM) and 132 GB disk, which is hosted on an Intel Core i5-8400 and employs the

Q35 machine type to facilitate USRP PCIe passthrough. The second VM, handling the Kubernetes control plane and 5G Core network functions, is hosted on an Intel Xeon Platinum 8470N and configured with the i440fx machine type, 4 vCPUs, 8 GB RAM, and a 32 GB disk. Finally, commercial Fibocom 5G modules serve as the User Equipment (UEs), featuring distinct Network Slice Selection Assistance Information (NSSAI) configurations to guarantee absolute slice isolation for each user.

We conducted an experimental evaluation of the dynamic RAN slicing capacity of the proposed virtualized architecture over a 60-second duration. The scenario comprised two UEs concurrently connected to the 5G network, each successfully attached to a distinct slice. To generate measurable data traffic and saturate the downlink, an `Iperf3` server deployed within the network core continuously transmitted data streams to both UEs simultaneously, aiming to achieve the highest possible throughput. During this period, we intentionally modified the Radio Resource Management (RRM) policies to evaluate the network's responsiveness by directly altering the maximum resource ratio parameters within the `rrmPolicy.json` file inside the gNB container. The experiment was structured into three distinct phases:

- **0s to 20s (Equilibrium):** Resources were evenly distributed, allocating 50% of the throughput capacity to Slice 1 (UE 1) and 50% to Slice 2 (UE 2).
- **20s to 40s (Priority to UE 1):** The policy was updated to prioritize Slice 1, allocating 55% of the resources to UE 1 and 10% to UE 2.
- **40s to 60s (Priority to UE 2):** The priority was inverted, shifting the allocation to 15% for Slice 1 and 30% for Slice 2.

Regarding network performance, as illustrated in Figure 3, the `Iperf3` logs confirm the throughput adaptation to the slice configurations. Initially, an equal resource distribution allowed both UEs to achieve approximately 60 Mbps. At the 20-second mark, a policy update slightly increased UE 1's throughput (via a 5% max ratio increment) while drastically reducing UE 2's rate to under 20 Mbps (as its max ratio was dropped from 50% to 10%). At 40 seconds, an inverted policy resulted in ~ 20 Mbps for UE 1 and ~ 40 Mbps for UE 2, a proportional performance that validates the newly assigned 15% and 30% slice ratios.

Concurrently, computational efficiency was evaluated using Telegraf (2-second resolution), tracking the RAN's `nr-softmodem` and the Core's `oai_*` microservices. The Central Processing Unit (CPU) metric aggregates multi-threaded usage, thus single-core percentages can exceed 100%. The plotted data in Figure 3 illustrates that CPU utilization is intrinsically tied to the aggregated resource allocation. As the sum of the max ratios across both slices decreased sequentially during the policy updates (from 100% to 65% and ultimately to 45%), the testbed's overall computational demand decreased correspondingly.

IV. CONCLUSION

This paper details the deployment of a cloud-native 5G testbed capable of dynamic RAN slicing and documents the

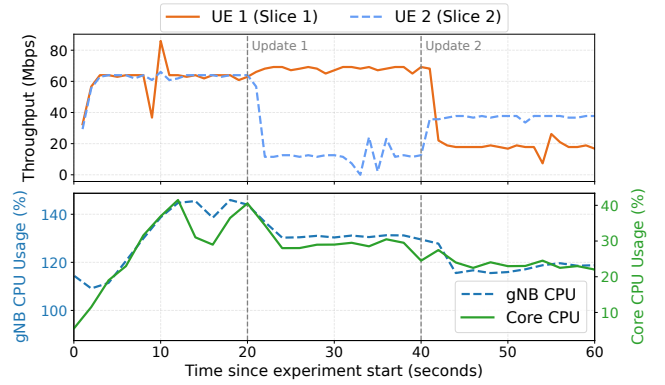


Fig. 3. Evaluation of the dynamic RAN slicing architecture. The top graph illustrates the downlink throughput (Mbps) achieved by UE 1 and UE 2 across three distinct RRM policy phases. The bottom graph correlates these network events with the underlying computational load, displaying the CPU utilization of the containerized gNB and 5G Core.

integration of RAN hardware into a dual-layered virtualized environment to address existing gaps in experimental methodologies. By employing PCIe passthrough alongside Kubernetes orchestration, our architecture satisfies the constraints of real-time signal processing while maintaining operational agility. The experimental validation confirms the system's capacity to perform dynamic RAN slicing, accurately adjusting throughput and computational resources on demand.

ACKNOWLEDGMENT

This study was financed in part by Ericsson Telecomunicações Ltda., Brazil; the Brasil 6G project (01245.020548/2021-07); Project Smart 5G Core And MultiRAN Integration SAMURAI (MCTIC/CGI.br/FAPESP under Grant 2020/05127-2); OpenRAN Brazil - Phase 2 project (MCTI grant N° A01245.014203/2021-14); and partially supported by the National Institute of Science and Technology (INCT) STREAM, funded by the Brazilian National Council for Scientific and Technological Development (CNPq), process no. 409179/2024-8.

REFERENCES

- [1] I. Afolabi *et al.*, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.
- [2] D. Johnson, D. Maas, and J. V. D. Merwe, "NexRAN: Closed-loop RAN slicing in POWDER - a top-to-bottom open-source open-RAN use case," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, 2021.
- [3] A. Kak *et al.*, "ProSLICE: An open RAN-based approach to programmable RAN slicing," in *IEEE Global Communications Conference (GLOBECOM)*, 2022.
- [4] P. Bertrand *et al.*, "SDN-based slicing testbed for 5G networks," in *2025 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2025.
- [5] L. Nóvoa *et al.*, "RAN slicing using OpenAirInterface and FlexRAN in a virtualized scenario," in *Anais do XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, 2020.
- [6] OpenAirInterface: 5G software alliance for democratising wireless innovation. OpenAirInterface Software Alliance. [Online]. Available: <https://www.openairinterface.org/>
- [7] H. Cheng *et al.*, "ORANSlice: An open-source 5G network slicing platform for O-RAN," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*, Washington, D.C., USA, Nov. 18–22 2024.