Deploying a Stable 5G SA Testbed Using srsRAN and Open5GS: UE Integration and Troubleshooting Towards Network Slicing

Conference Paper · August 2023

DOI: 10.1109/icABCD59051.2023.10220512

4 authors:

Lusani Mamushiane
CSIR
20 PUBLICATIONS **173** CITATIONS
SEE PROFILE

Albert A Lysko
Council for Scientific and Industrial Research, South Africa
119 PUBLICATIONS **546** CITATIONS
SEE PROFILE

Hlabishi Kobo
Council for Scientific and Industrial Research, South Africa
20 PUBLICATIONS **623** CITATIONS
SEE PROFILE

Joyce Mwangama
University of Cape Town
62 PUBLICATIONS **226** CITATIONS
SEE PROFILE

Deploying a Stable 5G SA Testbed Using srsRAN and Open5GS: UE Integration and Troubleshooting Towards Network Slicing

Lusani Mamushiane*,†, Albert Lysko *,† Hlabishi Kobo†, Joyce Mwangama*
†*Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa*
**University of Cape Town (UCT), Cape Town, South Africa*
[1]RVHLUS001@myuct.ac.za

***Abstract*—Field trials and experimentation are crucial for accelerating the adoption of standalone (SA) 5G in Africa. Traditionally, only network operators and vendors had the opportunity for practical experimentation due to proprietary systems and licensing restrictions. However, the emergence of open source cellular stacks and affordable software-defined radio (SDR) systems is changing this landscape. Although these technologies are not yet fully developed for complete 5G systems, their progress is rapid, and the research community is using them to test different use cases like network slicing. Building a 5G network is complex, especially in uncontrolled RF environments with fluctuating physical conditions such as noise and interference. This necessitates proper RF planning and performance optimization. The complexity is further compounded by the variety of 5G end-user devices, each with unique configurations and integration requirements. Some devices are network locked and require rooting to connect to a 5G testbed, while others need expert APN configurations or have specific compatibility specifications like sub-carrier spacing (SCS) and duplex mode. Unfortunately, vendors often provide limited information about RF compatibility, making trial-and-error techniques necessary to uncover compatibility details. This paper presents best practices for deploying and configuring a 5G SA testbed, focusing on the integration challenges of consumer-grade devices, specifically 5G mobile phones connected to a 5G testbed. Additionally, the paper offers solutions for troubleshooting integration errors and performance issues, as well as a brief discussion on the realization of basic network slicing in a 5G SA network.**

***Index Terms*—5G Standalone, 5G COTS UE, srsRAN, Open5GS, OpenAirInterface, Network Slicing**

I. Introduction

The fifth generation of mobile networks (5G) is expected to provide three generic services with radically different requirements, namely, enhanced mobile broadband (eMBB), massive machine-type communications (mMTC), and ultra-reliable low-latency communications (URLLC). Unfortunately, legacy network architectures were designed in a "one size fits all" manner, meaning that a single homogeneous network is used to serve all network services. This architecture design is not efficient in fulfilling the aforementioned service requirements. Laying down new network infrastructure for each separate service would be very expensive. A solution, known as network slicing, has been proposed by the Next-Generation Mobile Networks (NGMN) and Third Generation Partnership Project (3GPP) in TR23.799, to create multiple "fit-for-purpose" logical networks on top of a common physical network infrastructure. Network slicing capitalises on the flexibility of software-defined networking (SDN) and network function virtualisation (NFV) to deliver a more flexible network architecture through which each network service is allocated resources to provide performance guarantees and isolation from the other services. This approach is particularly appealing due to the inherent scarcity of external resources (power and spectrum) and infrastructural resources (compute, networking and storage), and provides a basis for infrastructure sharing among diverse ICT players ranging from virtual network operators (VNOs) to players that simply view connectivity as a service (such as Over-The-Top (OTT) service providers), and vertical industries. According to a projection by GSMA, the market for network slicing alone in the enterprise segment is said to be more than $300million$.

Commercially viable network slicing services require 5G standalone (SA) architectures, which, unlike the non-standalone (NSA) counterpart, incorporate a unique end-to-end slice identifier called the NSSAI, short for " Network Slice Selection Assistance Information". NSSAI consists of two sub-parameters, namely, the Slice/Service Type (SST), which can be either the eMBB, URLLC, or mMTC, and the optional Slice Differentiator (SD) to differentiate between slices with the same SST [1].

5G SA networks and user devices are becoming increasingly widespread, with more than 94 operators in 48 countries already investing in 5G SA networks [2]. At least 19 operators in 15 countries are understood to have launched commercial 5G SA networks, with the aim of offering new services more quickly, including those based on network slicing. However, the commercial uptake of 5G SA networks in Africa has been relatively slow, with most operators prioritising the deployment of 5G NSA. To the best of the author's knowledge, only Rain has launched 5G SA networks in Africa. This preference is mainly driven by the fact that 5G NSA guarantees a faster CapEx recovery by reusing existing 4G Core Networks, while 5G SA requires investment in a new, costly Core Network.

The importance of field trials and experimentation to accelerate the commercial uptake of 5G SA in Africa cannot be overstated. Due to the proprietary nature of cellular systems and licencing restrictions, practical experimentation with

TABLE I: Feature_based comparison between srsRAN and OpenAirInterface

Feature	**srsRAN**	**OpenAirInterface**
Frequency Bands Supported	All FR1 (sub-6 GHz) bands	All FR1 and FR2 bands
Duplex Mode	FDD/TDD	FDD/TDD
Sub-Carrier Spacing	15/30KHz	15 and 30kHz (FR1), 120kHz (FR2)
Bandwidth (BW)	10, 20, 40, 80, 100MHz	10, 20, 40, 80, 100MHz
Protocol Stack Implemented	Full stack	Full stack
Programming Language	C & C++	C & C++
Tested performance	64 simultaneous users Speed achieved: 140 Mbps DL and 120 Mbps UL Round trip time (RTT): Not specified	16 simultaneous users Speed achieved: 1Gbps DL and 100 Mbps UL RTT: 4ms
Hardware Requirements	Intel-based x86 PC with at least 4 CPU cores running at least at 2.7GHz	Intel based-x86 PC with at least 4 CPU cores running at least at 2.7GHz
Antenna Configuration	SISO	SISO and MIMO
Support Documentation & Community Support	Excellent	Fair
Operating System (OS)	A standard Linux -based OS	A standard Linux -based OS
Contributors	98	105
Licensing Model	GNU Affero General Public License v3.0 [1]	OAI Public License [2]
Deployment Complexity	Easier and faster to deploy	Complex to deploy

TABLE II: Comparison of projects focused on 3GPP-compliant 5G Core Network implementations

Project	**Open Source?**	**3GPP Compliance**	**Production Grade**	**Deployment Complexity**	**Cloud-native?**	**License**	**Number of Contributors**	**Hardware Requirements**
Magma [3]	Yes	Release 16	Academic product	Easy to deploy and use	No	Apache-2.0	10	6GB RAM, 2 CPUs, 60GB HDD
Open5GS [4]	Yes	Release 17	Semi-academic product	Fast and easy to deploy	Yes	AGPL	66	4GB RAM, 1 CPU, 20GB HDD
Free5GC [5]	Yes	Release 15	Semi-academic product	Easy to deploy and use	Yes	Apache-2.0	30	4GB RAM, 1 CPU, 30GB HDD
OAI 5GC[6]	Yes	Release 16	Academic product	Easy to deploy, difficult to use	Yes	OAI Public License	Not clear	4GB RAM, 2 CPUs, 40GB HDD
OMEC [7]	Yes	Release 16	Semi-academic product	Challenging to deploy and use	Yes	Apache-2.0	24	4GB RAM, 1 CPU, 20GB HDD

mobile networks has historically been reserved exclusively for network operators and vendors. This is changing with the emergence of open source cellular stacks and affordable software-defined radio (SDR) systems. Although these have not yet matured enough to underpin complete 5G systems, their development is progressing quite rapidly, and the research community has started experimenting with these open-source systems to test various 5G applications, such as network slicing.

Several free and open source 5G SA cellular platforms have emerged in the past few years. These platforms provide an opportunity for the research community to join forces with the industry in practical experimentation efforts and foster 5G SA standardisation and optimisation efforts. There exist three popular free and open-source 5G radio access network (RAN) projects, namely, srsRAN[8], OpenAirInterface (OAI)[9], and UERANSIM[10], available for exploration by the research community. Both srsRAN and OAI implement a subset of 3GPP Release 16. Unlike srsRAN and OAI, UERANSIM only implements Layer 3 radio protocols (constituting the RRC and NAS layers) and does not implement Layer 1 (i.e., PHY) and Layer 3 (i.e., MAC, RLC, and PDCP) of the RAN protocol stack. On the contrary, srsRAN and OpenAirInterface provide the full version of the complete protocol stack in accordance with the 3GPP standards and specifications for 5G networks. Table I provides a feature-based comparison of these platforms. The key difference between the two platforms is in the number of sub-carrier bandwidth (numerology), number of channels (whether Single-Input Single-Output(SISO) or Multi-Input Multi-Output (MIMO)), and support documentation. OAI provides more flexibility in the sub-carrier bandwidth, which, for higher bandwidths, enables lower latency and support for higher-frequency (mmWave) bands. While srsRAN can only work with SISO antenna configurations, OAI supports both SISO and MIMO channels, thus providing better throughput under the same signal-to-noise-plus-interference (SNIR) conditions. Evidently, OAI is much more mature in terms of its feature base and its user base. However, based on our experience experimenting with both platforms, srsRAN provides better community support and is well-documented, making it easier to deploy. For these reasons, we recommend srsRAN as a starting point toward 5G network research and capability development.

There are also several notable projects focussing on 3GPP-compliant 5G core network implementations, such as Magma [3], Open5Gs [4], Free5GC [5], OAI 5GC[6], and OMEC [7]. Table II provides a comparison of these implementations based on supported features. The OAI core network is part of the OAI 5G ecosystem and, like Magma, is geared towards academic environments for proof-of-concept demonstrations. The other core network implementations are approaching production readiness, with Open5GS having been validated through a joint collaboration between Cloudify, Intel, CapGemini and AWS [11]. Core networks are lightweight and are less compute resource hungry than RAN systems. For this reason and depending on available resources, the core network can be deployed in a virtualised environment or containers without any significant performance degradation.

II. Contribution

Deploying a stable 5G SA testbed to inter-work with consumer-grade user equipment (UE), such as a 5G mobile phone or modem, is not a trivial undertaking, as it involves integrating disparate, inherently complex software stacks, RF tuning and optimisation experiments. This paper presents our 5G testbed deployment fully constructed using an open source 3GPP-compliant cellular stack and core network on commercial off-the-shelf (COTS) computing environments. In particular, we build our testbed leveraging the srsRAN and Open5GS solutions. We opted for the implementation of the Open5G core network largely due to its coverage for adoption, wide community support, and support for the latest 3GPP specifications, compared to its rivals. The decision to use srsRAN was partly due to its community support, stability, and modular codebase (making it easier to customise compared to OAI). Based on our experimental exploration of OAI, we found it to be slightly unstable with poor community support. To date, there have been several research efforts (such as [12], [13], [14], [15], [16], [17], [18], and [19] that focus on demonstrating open source driven 4G and 5G testbed deployments. However, most of the existing works are either based on 4G or 5G NSA testbed deployments and do not fully expose the complexity of integrating a real UE to a 5G SA network. For example, Ssonko et al. [17] provide a 5G testbed deployment guide to implement a device-to-device (D2D) communication network leveraging a srsRAN-based 4G network. Moreover, most of the existing works focus on 5G network performance profiling with no RF optimisation efforts. Chun et al. [13] quantify the performance of an OAI-based 4G network by performing throughput profiling using an emulated air and UE interface. Similar to [13], Gringoli et al. [14] assess the performance of a srsRAN-based 4G network and an OAI-based 4G network and benchmark the throughput achievable by each software stack using a real mobile phone. Chepkoech et al. [15] present a useful guide to configuring, deploying, operating, and evaluating the performance of open source based 4G and 5G (SA and NSA) mobile network testbeds. The authors successfully tested consumer-grade 5G UEs (namely, a mobile phone, a router, and a Pi Hat). The only shortcoming of their work is that they did not divulge details on UE integration challenges, troubleshooting, and network slicing implementation details. Our work in [18], provides some building tips and resource consumption footprint of an OAI-based 5G NSA testbed evaluated using a 5G mobile phone. The shortcoming of our work is that we limited our scope to the 5G NSA deployment mode. Furthermore, the integration of 5G UE was easy for this mode. As such, we did not ponder on UE integration considerations. Mihai et al. [16], conduct performance benchmarking of 5G SA networks based on OAI and srsRAN cellular stacks by assessing performance metrics of throughput and latency. Like other works, these authors do not describe the minute details of their testbed implementation and UE integration procedures and challenges. Fabian et al. [19] provides valuable performance results of an OAI-based 5G SA network using a Quectel RM500Q-GL cellular modem as the UE. This work is based on an interference-free RF interface through the use of SMA coaxial cables between the UE and the RAN system. The authors superficially highlight the steps taken to integrate the 5G UE into their network. Our study attempts to close this gap by describing the UE integration procedures and complexity in a verbose way. In summary, the key contribution of our study is threefold:

- It provides a deployment guideline on setting up an end-to-end 5G network, with more focus on the key deployment challenges, building tips, and troubleshooting techniques.
- It sheds light on the integration challenges of a consumer-grade 5G UE to a 5G testbed and offers RF configuration guidelines.
- It provides practical insights into network slice setup from the UE's perspective and enforcement on each network domain (i.e. RAN and Core).
- Last but not least, this paper provides decision-making criteria for selecting an appropriate UE for integration into a 5G network.

A. *Organisation of Paper*

The rest of this article is organised as follows. In the next section, we describe our testbed setup, and best practises for configuring the different network domains (i.e. RAN, backhaul, and Core Network). We further shed light on the integration challenges of consumer-grade UEs, particularly a 5G mobile phone connected wirelessly without RF shielding. Subsequently, we provide solutions to troubleshoot various integration errors and performance issues. Finally, we briefly discuss network slicing enforcement and message follow on both consumer-grade UE devices.

III. Testbed Setup, Deployment, Configuration, and Validation

This section provides a deployment guideline for a functional 5G prototype, including the error cause values as well as troubleshooting tips.

A. Testbed Setup

As depicted by Figure 1, our testbed comprises the srsRAN cellular stack deployed on a commodity Linux-based compute node (Intel x86 PC architectures) and a software-defined radio (that is, a NI-2944R universal software radio peripheral (USRP)) from NI Instruments. The srsRAN stack serves as the baseband processing unit (BBU), while the USRP serves as the remote radio head (RRU) responsible for receiving, transmitting, filtering, and amplifying RF signals. We opted for a USRP as it allows reconfigurability and programmability of certain functionalities (radio protocols, operating frequencies, modulation method, gain, etc.) to suit prevailing conditions (such as interference, noise, and traffic volume). We configured the USRP to operate in SISO mode and without beamforming, which results in moderate throughput for a 5G network. To achieve a close to carrier-grade private 5G network, we recommend MIMO antenna configurations with beamforming for increased data capacity.

The fronthaul connection (i.e. between the srsRAN and USRP) is via a PCIe connector to achieve lower latency and higher data transfer. The Core Network is deployed leveraging the service-based Open5GS running in an OpenStack virtual environment. Each Open5GS service is configured to run on a Docker container and communicate over a private network. The backhaul connection (i.e., between the BBU and the Core Network) uses a 1GB Ethernet switch. Although the focus of this paper is Open5GS, our testbed comprises many other implementations of the 5G and 4G core networks (some of which are described in Table II) hence the use of a switch in the backhaul. To tune the proposed 5G testbed towards high performance with maximum throughput and low latency, we recommend the use of a 10 GB backhaul link to achieve higher network data rates. The Internet breakout is through a local area network to a "fiberised" last-mile network. To date, we have experimented with different consumer-grade UE models (namely, Xiaomi Redmi Note 11 Pro, Huawei P40 Pro, Samsung S21, and a Huawei 5G Router CPE) and emulated UE stacks (namely, srsUE, UERANSIM and openAirInterfaceUE). Currently, only a handful of UEs are available that are compatible with 5G SA on the market. Most UEs that claim compatibility with 5G work only with 5G NSA networks. Of the consumer grade UEs with which we experimented, we were able to successfully connect the Huawei P40 Pro and Samsung S22 to the 5G testbed. By virtue of being emulated, platforms designed and optimised to work with 5G SA, UERANSIM, srsUE, and openAirInterfaceUE are seamlessly integrated into the testbed. For this reason, our paper will be largely centred around the UE (Huawei P40 in particular) that worked well with our testbed. The UE connects to the Internet via a User Plane Function (UPF) over a masqueraded GTP tunnel. Table III lists all components that we used to deploy our 5G SA testbed.

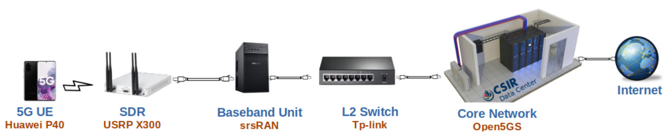


Fig. 1: Experimental setup

TABLE III: Hardware Specification for the Testbed Deployment

Component	**Specification**
User Equipment (UE)	
Mobile Phone	Huawei P40 pro
Android Version	Android 10
Supported 5G SA bands	1, 3, 28, 38, 41, 77, 78, 79
SIM Card	SysmoISIM-SJA2
Software Defined Radio (SDR)	
Vendor	National Instruments
Model	NI-2944R
Fronthaul Interface	PCIe
Number of Channels	4 (2XTx and 2XRx)
Frequency Range	10 MHz to 6 GHz
Bandwidth	160MHz
Gain Range	0 dB–31.5 dB (Tx) 0 dB–37.5 dB (Rx)
Baseband Unit (gNB)	
Computer	Dell Precision 3630
CPU	IntelCore™i9-9900 CPU @ 3.10GHz×16
Memory	64GB
Operating System	Ubuntu 20.04, x86_64 bit
Storage	300GB SSD
Kernel	5.4.0-147-lowlatency
Ethernet Switch	
Vendor	Tp-link
Type	8-port Gigabit Desktop Switch
Ports	8
Maximum speed	1Gbps
Core Network	
Computer	OpenStack Virtual Machine
vCPU	2
Memory	4GB
Operating System	Ubuntu 20.04
Storage	30GB

B. RAN Deployment: srsRAN gNB/BBU

Before running the srsRAN software, it is strongly recommended to tune the compute node to the optimal performance mode to meet the stringent USRP's performance requirements for real-time communication. It is important to avoid running the software on a virtual machine by installing it directly on the metal. Performance mode is achieved by implementing the following settings:

- Setting the CPU governor to performance – for a more responsive RAN and minimal packet drops.
- Disabling HyperThreading – to reduce the latency of the CPU.
- Disabling virtualisation technology – to enhance stability and performance.
- Enabling the thread priority scheduling – to reduce real-time overhead and increase system throughput.
- Setting the socket buffer sizes – to reduce packet drops due to buffer overflows [20].
- Setting Ethernet MTU values – by default, the MTU

value for a 5G cellular channel interface is 1500 bytes. Depending on the channel capacity, a higher value is recommended to increase the data transfer rate.
- Setting network interface card (NIC) ring buffer sizes – to lower the number of interrupts sent to the CPU. The maximum buffer size varies depending on the NIC installed. Intel NICs typically cap the size at 4MB for 1GB NICs.
- Disabling C-state support – prevent the RAN from entering sleep state and hibernation mode.
- Configure a low-latency kernel – to reduce overhead while maintaining responsiveness.
- Optionally, the Data Plane Development Kit (DPDK) can be installed and activated to accelerate CPU performance. However, this is not necessary to run any of the FR1 channel bandwidths.

We use the script in [21] and the tutorial [22] for our performance settings. After tuning the system to optimal performance mode, the next step is to install the srsRAN dependencies and the USRP driver (i.e., UHD). We recommend building UHD from source instead of installing from a binary package, as it enables flexibility in upgrading and downgrading versions and allows users to modify the codebase and create customised versions. Importantly, the USRP should not be connected to the system during UHD installation. To verify the installation, connect the USRP, restart the server, and run "uhd_find_devices" and "uhd_usrp_probe". The full set of instructions for UHD installation is available at [23].

The next step is to clone the srsRAN repository and build the codebase as described in the srsRAN installation guide[24]. In order to analyse the logs on different layers of the radio protocol stack (such as MAC, RLC, RCC, etc.), it is important to enable the level of logging as part of the system configuration [25].

To troubleshoot USRP-related issues, users should always ensure that they have loaded UHD FPGA images using "uhd_images_downloader" and ensure the USRP they are using is correctly configured and running over USB 3.0 (for B210 USRPS), Ethernet, or PCIe. If the UHD installation was from source code, users could verify the USRP configuration by running the benchmarking application available in the "examples" folder typically located in "/usr/lib/uhd". If the issues are related to PRACH transmissions, users need to align the USRP and BBU time calibration according to the srsRAN manual [26].

C. Core Network Deployment: Open5GS

The Core Network can be deployed on an Ubuntu OS running directly on the metal or a virtual machine (KVM, VMWare, or VirtualBox hypervisor). The Core Network is not as resource intensive as the RAN and can (a for minimal prototype) be deployed on a Raspberry Pi as long as it meets the minimum hardware requirements specified in Table II. However, for a closer to carrier-grade deployment, we recommend a higher-end computing node to be able to demonstrate use cases such as massive machine-type communications requiring high-performance user plane functions (UPF).

To deploy Open5GS, the first step is to install the database for 5G network functions, namely the NF repository function (NRF), the policy control function (PCF), and unified data management (UDM). This is followed by the installation of dependencies and setting up of the tunnel interface for user traffic routing. The tunnel interface is not persistent after rebooting and should always be created before running the Core Network services. We recommend creating a script to set up the tunnel on reboot. Once all dependencies are installed, the next step is to clone the Open5GS repository and build the software from the source, by following the instructions in [27]. Basic 5G functionality is provided by the Access and Mobility Function (AMF), Session Management Function (SMF), UDM and UPF, meaning that a user can deploy the said functions for a minimal prototype. To provision network subscribers in the database, we recommend installation of the web user interface, specially designed to accommodate novice Open5GS users.

To avoid firewall rules that block traffic, it is important to disable the firewall. If the Core Network services fail to run, the user must ensure that the tunnel interface is active using "sudo ifconfig name_of_interface up". If the interface is up and the services still fail to run, then the user must confirm that all Core Network processes are shut down and kill processes that are still running.

D. User Equipment (UE) Deployment

There are three scenarios for the UE implementation: using a USRP radio; a 5G wireless modem module; or using a commercial-off-the-shelf (COTS) handset. The USRP scenario uses an emulated UE deployed on a COTS server and a USRP serving as an antenna to complete the UE stack. srsRAN project includes an implementation of the UE stack dubbed srsUE. Like srsRAN, OAI has implemented a 5G UE stack called OpenAirInterfaceUE. These stacks are cross-compatible with various USRP radio models. Importantly, an OctoClock-G device is needed to synchronise the gNB USRP and the UE USRP. The "-G" model is shipped with an internal GPSDO module and is thus recommended for the USRP-based UE deployment.

The procedure for installing the srsUE software is similar to the procedure for installing the srsRAN gNB software, except that the building process only includes the UE component. An important consideration during system build is to build the UE on a dedicated compute node for resource isolation and optimisation purposes.

For deployments using a 5G wireless modem module, we recommend the Quectel RM500Q-GL 5G wireless modem. The Quectel modem is a 5G new radio (NR) sub-6GHz module fully optimised for industrial IoT and eMBB applications. It complies with 3GPP Release 15 specifications and supports both 5G SA and NSA configurations. It is entirely possible to integrate other modem modules as well, such as the Sierra Wireless EM9191 module.

In our case, we experimented with COTS UEs such as Xiaomi Redmi Note 11 Pro, Huawei P40 Pro, Samsung S21, and a Huawei 5G Router CPE) as well as USRP-based stacks using srsUE and OpenAirInterfaceUE. This paper focuses on our experience using the Huawei P40 handset. Notably, the USRP-based UE implementation does not require a SIM card. A "programmable" SIM card supporting XOR/MILENAGE algorithm with known keys is required for the wireless module and handsets. To test the 5G testbed using a handset or modem, an external clock source such as an Octoclock or GPSDO (compatible with the USRP used) is strictly required. This is primarily because the on-board clock within the USRP may not be accurate enough to enable a connection with the UE. Our testbed uses a GPSDO reference clock.

When configuring the gNB, it is important to use a 30kHz sub-carrier spacing (SCS) for TDD bands. This is because many commercial COTS UEs require a 30kHz SCS for TDD configurations. The only configurations required on the UE are SIM card programming and access point name (APN) settings, which should correspond with the Core Network APN (i.e. DNN in 5G) settings. The APN can be set to the Internet, IP multimedia subsystem (IMS), or any other 3rd party APN. Our testbed is currently configured to use the Internet APN.

1) SIM Card Programming: Different COTS SIM card models are available for use with 5G testbeds. We recommend SIM cards from Sysmocom (such as SJS1 and SJA2)[28] and Open-Cells [29]. These SIM cards support 3GPP standards and specifications and support the 3GPP MILENAGE authentication algorithm. We tested our testbed using both Sysmocom and open-Cells cards. The cards were reprogrammed to align the authentication and operator-specific parameters (IMSI, OPC, Ki, etc.) with the Core Network configuration. The Sysmocom cards were programmed using PySim [30] — a SIM card reader and writer utility. When using 5G-enabled sysmcom-ISIMs, it is important to enable SUCI concealment following the guide by [31].

2) Air Interface: An RF license is required to connect to the testbed over the air interface. In the absence of a license, an RF Enclosure/Faraday Cage may be used to avoid interference with licensed network operators. To sanitise the air interface from too much noise and interference, a direct connection to the USRP can be made using an SMA coaxial cable and RF splitters. According to [32], the transmission over a coaxial cable is more resilient to external noise, and so the SNIR is better than transmitting over wireless.

E. End-to-end 5G Connection Establishment

To establish an end-to-end connection, the PLMN, NSSAI (SST and SD) corresponding to the Core Network settings must be configured. The selected PLMN values must be free and not reserved by mobile network operators. The Core Network should be reachable from the gNB and the UE. The data and control message exchange between the gNB and Core Network is over a private network, while the communication between the UE and Core Network is over the gNB private network for control traffic and GTP tunnel for data traffic. In 5G, a successful GTP tunnel established is indicated by a successful "PDU Session Establishment Response" message from the Core Network. Once a PDU session has been established, the UE can start consuming Internet services. To bridge between the Core Network and the Internet, it is important to enable IP forwarding and add NAT rules to IP Tables. Otherwise, the UE will not be able to receive Internet traffic. In case the UE cannot see the network, Network Signal Guru (available on Play Store), can be used to force the network to see the 5G SA cell. For this to work, the device must have a Qualcomm baseband processor and also be rooted. The full instructions on how to configure this are available at [33]. Additionally, the transmit (Tx) and receive (Rx) gain values can be adjusted if the UE is still unable to see or connect to the network. Adjusting the gains can also improve the performance of the network throughput. Tables IV and V, provide troubleshooting techniques applied to achieve a stable 5G SA testbed.

F. Enabling Network Slicing on UE

Network Slice Selection Assistance Information (NSSAI) is a unique end-to-end slice identifier signalled by end-user devices to assist the network in associating a particular network slice with a UE. NSSAI is managed at the tracking area level in the gNB, and at the registration area level in the Core Network. There are different types of NSSAIs (see Figure 2, such as Configured NSSAI (provisioned in the device), Subscribed NSSAI (NSSAI stored in the UDM), Requested NSSAI (signalled by the UE during the registration procedure) and Allowed NSSAI (the NSSAI that the UE can use for the current registration) [1]. The Requested NSSAI is determined by the UE at run-time by a relatively complex procedure. It is not certain ahead of time and is not read directly from the UE configuration files. In the Core Network, these NSSAIs are provided by the NSSF when queried by the AMF network function.

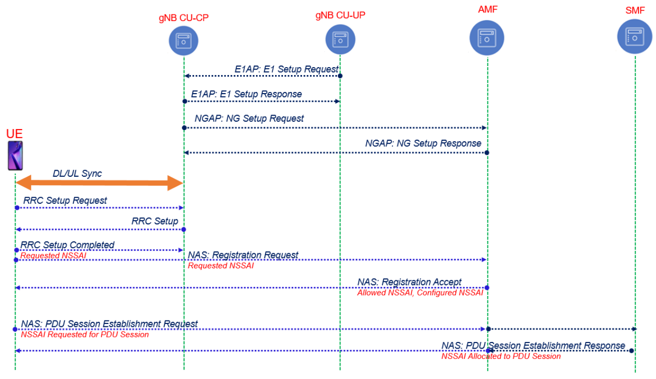


Fig. 2: NSSAI message exchange illustration

For cases where a Configured NSSAI is used, a UE can request multiple slices, such as eMMB and URLLC, by passing different SST values. Within a network slice, UEs can create PDU sessions to different gateways via DNN. Within a PDU session, the UE can establish multiple quality of service (QoS) flows (EPS bearers in LTE) with specific service characteristics.

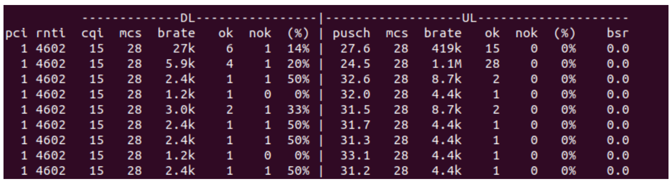
An important criterion to consider when choosing a handset capable of network slicing is the Android OS version supported by the device. For devices running Android 12 or higher, Android provides support for 5G network slicing using a new dynamic policy control mechanism called the User Equipment Route Selection Policy (URSP) technology, instead of setting up slices through APNs. URSP is enabled by the Core Network's policy control function (PCF) which provides URSP rules to the UE via the AMF. In late 2022, Google (in partnership with Ericsson) conducted a trial to validate multi-slice support by a Google Pixel 6 Pro handset running their newly released Android 13 OS. The trial showed that a single handset running the Android 13 OS could connect to multiple network slices simultaneously depending on whether the application running is an enterprise or consumer application. For example, a handset can connect to a low latency slice for video streaming while at the same time connecting to a best-effort slice for regular messaging, and connect to a secure and high-performance slice for sending and receiving business-sensitive information. This means that application developers can now request what connectivity category (latency or bandwidth) their application will need, and then an appropriate slice, whose characteristics are defined by the mobile network, will be selected.

Android 13 enables up to eight slice categories (including enterprise and consumer slices) to be associated with the handset profile [34]. Unfortunately, apart from Google's own Pixel range of devices, Android 13 is not available on any other Android handsets, and according to [35], the implementation of Android 13 on handsets is expected to be slow and fragmented. Our Huawei P40 Pro handset is running EMUI 10.1 (based on Android 10), and as such, it neither supports NSSAI pre-configuration, automatic switching between network slices, nor sends the Requested NSSAI message to the Core Network. Instead, the handset waits for the AMF to send the Allowed NSSAI during the PDU setup procedure, which it then connects to and stores in its database for subsequent connections.

1) NSSAI Storage: According to [36], the UE stores allowed NSSAI until it received a request from the network to remove the NSSAI due to changes in the network slice subscription. In our case, Huawei P40 always needs to be rebooted in order to connect to the network using new NSSAI configurations. Otherwise, the phone failed to connect, citing a "DNN Not Supported or Not Subscribed in the Slice" error. Our NSSAI setting was for eMBB with SST=1 and SD=0. This was because, at the time of testbed deployment, srsRAN only supported an SST=1 with a flexible SD configuration.

IV. Performance Measurement

Figure 3 shows the gNB trace of the Huawei P40 sending and receiving data over our testbed with minimal RF optimisation (optimum Rx and Tx gain and attenuation, 30kHz SCS, and 100 resource blocks (RB), etc.). Results related to modulation and coding scheme, MIMO, noise, interference, and channel bandwidth optimisations will be reported in our future work. From the console below, "cqi" is the channel quality indicator reported by the UE. The UE estimates the CQI based on radio conditions (noise and interference) and ranges from 1 to 15, with 15 indicating perfect radio conditions. "ok" denotes the number of packets successfully sent, "nok" is the number of packets dropped, and "brate" denotes the bit rate measured in bits/sec.



```
               -------------DL---------------|------------------UL-------------------
pci rnti  cqi  mcs  brate   ok  nok  (%) | pusch  mcs  brate   ok  nok  (%)    bsr
  1 4602   15   28    27k    6    1  14% |  27.6   28   419k   15    0   0%    0.0
  1 4602   15   28   5.9k    4    1  20% |  24.5   28   1.1M   28    0   0%    0.0
  1 4602   15   28   2.4k    1    1  50% |  32.6   28   8.7k    2    0   0%    0.0
  1 4602   15   28   1.2k    1    0   0% |  32.0   28   4.4k    1    0   0%    0.0
  1 4602   15   28   3.0k    2    1  33% |  31.5   28   8.7k    2    0   0%    0.0
  1 4602   15   28   2.4k    1    1  50% |  31.7   28   4.4k    1    0   0%    0.0
  1 4602   15   28   2.4k    1    1  50% |  31.3   28   4.4k    1    0   0%    0.0
  1 4602   15   28   1.2k    1    0   0% |  33.1   28   4.4k    1    0   0%    0.0
  1 4602   15   28   2.4k    1    1  50% |  31.2   28   4.4k    1    0   0%    0.0
```

Fig. 3: srsRAN gNB trace for our setup

V. Conclusion

This paper provided an empirical guideline on deploying a 5G standalone testbed leveraging open source software stacks, namely srsRAN and Open5GS. The paper provides researchers interested in building an indoor testbed with tips on how to rapidly realise a basic 5G prototype. Best practises for configuring the different network domains (i.e., RAN, backhaul, and Core Network) were also provided. Furthermore, this paper sheds light on the integration challenges of consumer-grade UEs, particularly a 5G mobile phone connected over wireless without RF shielding. Subsequently, we provide solutions to troubleshoot various integration errors and performance issues. Finally, we discuss network slicing enforcement and message follow on both consumer-grade and emulated UE devices through analysis of packet traces over our testbed.

References

[1] "Network slice selection function (NSSF) cloud native user's guide," 2020, Last accessed 15 March 2023. [Online]. Available: https://docs.oracle.com/

[2] "5G Standalone Market Report," 2021, Last accessed 01 April 2023. [Online]. Available: https://gsacom.com/paper/5g-standalone-november-executive-summary/

[3] "Magma," 2018, Last accessed 28 February 2023. [Online]. Available: https://github.com/magma/magma

[4] "Open5gs," 2017, Last accessed 03 March 2023. [Online]. Available: https://github.com/open5gs/open5gs

[5] "Free5GC," 2019, Last accessed 30 January 2023. [Online]. Available: https://github.com/free5gc/free5gc

[6] "OpenAirInterface 5GC," 2019, Last accessed 05 February 2023. [Online]. Available: https://gitlab.eurecom.fr/oai/cn5g

[7] "OMEC UPF," 2020, Last accessed 03 March 2023. [Online]. Available: https://github.com/omec-project/upf

[8] "srsRAN Documentation," 2018, Last accessed 13 April 2023. [Online]. Available: https://docs.srsran.com/en/latest/

[9] "OpenAirInterface," 2018, Last accessed 13 April 2023. [Online]. Available: https://openairinterface.org/

TABLE IV: Troubleshooting guideline for a 5G SA Testbed (Part A)

Issue	**Bottleneck**	**Possible Cause**	**Recommended solution**
Crashing gNB (sometimes throwing a segmentation fault).	RAN	-Active sleep and hibernation modes. -gNB software running in debug mode. -Too many background processes running not needed by the gNB.	(1) Set Linux to run in performance mode. (2) Try to avoid running the gNB in debug mode to minimise disk I/O operations. (3) Sanitise the Linux environment by uninstalling applications not needed to run the gNB and disabling auto-updates.
PDN connectivity reject (unknown or missing APN).	Core Network	-UE does not support SA on test PLMN resulting in 5G being disabled due to policy violations. -Most 5G handsets typically contain carrier policies which do not permit connection to test PLMNs.	(1) Using an RF shield and configuring the network with a commercial network PLMN. (2) Using any other PLMN other than 00101 and 99999.
Phone not seeing or connecting to the network.	RAN and Core Network	5G SA not enabled or inaccurate clocks at the gNBs RF frontend.	(1) On some handsets, when using a test USIM/ISIM, 5G SA can be activated using %*#*#4636#*#*. (2) Rooting the UE and using Network Signal Guru to force the UE to see the 5G cell \cite{cots21334}. (3) Use an external 10 MHz reference or use a GPSDO-disciplined oscillator. (4) Always toggle to flight mode before each connection attempt. (5) Make sure to set the sub-carrier spacing to at least 30kHz. (6) Switch between FDD and TDD to see what is compatible with UE under test. (7) Make sure the USRP is properly time calibrated.
Low throughput	RAN	-srsRAN new radio (NR) schedular does not support dynamic MCS adaptation. -Instead MSC is hard coded to 28.	(1) Use lower MCS values suitable for current RF conditions. (2) Confirm UE modulation method using Network Signal Guru utility and use the results to set an appropriate MCS value. (3) Assign more physical resource blocks. (4) Kill unnecessary background processes. (5) Check the physical orientation of the antennas. (6) The throughput is limited with the USRP B210 devices without amplification and MIMO therefore, using a higher-end USRP is recommended.
Too many UUUs and LLLs on gNB stdout.	RAN	CPU not configured to run on performance mode.	(1) Configure CPU to performance mode. (2) Confirm if the sampling rate is set correctly. (3) Kill all unnecessary processes. (4) Check the USB3 connection (some cables are bad).
DNN not Supported or not Subscribed in the Slice.	Core Network	-DNN incorrectly configured on Core Network or subscriber database. -NSSAI (SST and SD) incorrectly configured on either UE or Core Network. Android Version not supported or incompatibility issues with 5G SA.	(1) Confirm DNN and NSSAI settings on Core Network and database. (2) Check Android Version compatibility with 5G SA network configuration.

TABLE V: Troubleshooting guideline for a 5G SA Testbed (Part B)

Issue	Bottleneck	Possible Cause	Recommended solution
UE not connecting to the Internet.	Core Network	Typically caused by DNN misconfigurations or an inactive IPv4 forwarding.	(1) Make sure the DNN settings on UE and SMF are the same i.e., the DNN name and protocol family (IPv4/IPv6). (2) Enable IP packet forwarding.
Invalid/unverified signatures during MongoDB installation after executing "apt-get update".	Core Network	MongoDB Public GPG key was not added during the import step.	(1) Run: "sudo rm /etc/apt/sources.list.d/mongodb*.list". (2) "sudo apt-key adv –keyserver keyserver.ubuntu.com –recv-keys <PUBKEY>".
Uplink failure timer timeout causing NGAP UE context to be removed.	RAN	Usually caused by poor radio conditions resulting in a high signal-to-interference-plus-noise ratio (SINR).	(1) Verify root cause by using a Spectrum Analyser to estimate SINR. (2) Lower the attenuation of the Rx and Tx antenna on the USRP Radio head by 2 until the problem is resolved. (3) Tuning the PUSCH and PUCCH target powers through trial and error. (4) Reduce the Tx gain and increase the Rx gain.
UE Authentication failure.	Core Network	Parameters (OPc, K, SQN) programmed on the SIM card mismatch the parameters provision on database.	Verify that SIM card details and compare with database entries.
Illegal UE	Core Network	IMSI are not correctly configured.	(1) Make sure the PLMN on the SIM card matches the network PLMN. (2) Make sure the IMSI on the SIM card matches the database entry for the UE.
Invalid NR band	RAN	srsRAN only supports bands in 3GPP Release 15.	Make sure to use a band in Release 15, i.e. (FR1), which extends from 450 MHz to 7.125GHz, and frequency range 2 (FR2), which extends from 24.25 to 52.6GHz.
Data transmission does not work during second UE connection attempt.	RAN	At the time of writing RRC Reestablishment had not yet been implemented.	Restart UE and test connection again.
No NGAPSetupResponse	Core Network and RAN	Incorrect configuration of parameters needed to connect the gNB.	(1) Make sure the firewall is disabled on RAN and Core Network server. (2) Make sure NSSAI and TAC in RAN matches configurations in Core Network. (3) Make sure all critical network functions (AMF, UPF, UDM, and SMF) are running.
External GPSDO error "Could not find sync source".	RAN	External reference cannot be both a clock source and a time source.	(1) Make sure to set only clock to external and don't select sync source.
Late, overflow and underflow error.	RAN	User doesn't have privileges to run real-time thread.	Grant user privileges to run real-time threads.
gNB not connect to Core Network.	RAN	Incorrect gNB configurations.	(1) Check IP addresses in config files. (2) Check PLMN and TAC values for correspondence with Core Network settings.
UE c689 not found.	RAN	This typically happens when the maximum Rx gain is too high.	Reduce Rx gain value to a lower number in intervals of 5 until the error is eliminated.

[10] “UERANSIM,” 2020, Last accessed 13 March 2023. [Online]. Available: https://github.com/aligungr/UERANSIM

[11] “End-to-End 5G Network Slicing with Cloudify, Intel and CapGemini on AWS Infrastructure,” 2021, Last accessed 10 April 2023. [Online]. Available: https://talks.cloudify.co

[12] T. Dreibholz, “Flexible 4g/5g testbed setup for mobile edge computing using openairinterface and open source mano,” in *Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 34th International Conference on Advanced Information Networking and Applications (WAINA-2020)*. Springer, 2020, pp. 1143–1153.

[13] Y. Y. Chun, M. H. Mokhtar, A. A. A. Rahman, and A. K. Samingan, “Performance study of lte experimental testbed using openairinterface,” in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, 2016, pp. 617–622.

[14] F. Gringoli, P. Patras, C. Donato, P. Serrano, and Y. Grunenberger, “Performance assessment of open software platforms for 5g prototyping,” *IEEE Wireless Communications*, vol. 25, no. 5, pp. 10–15, 2018.

[15] M. Chepkoech, N. Mombeshora, B. Malila, and J. Mwangama, “Evaluation of open-source mobile network software stacks: A guide to low-cost deployment of 5g testbeds,” in *2023 18th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2023, pp. 56–63.

[16] R. Mihai, R. Craciunescu, A. Martian, F. Y. Li, C. Patachia, and M.-C. Vochin, “Open-source enabled beyond 5g private mobile networks: From concept to prototype,” in *2022 25th International Symposium on Wireless Personal Multimedia Communications (WPMC)*. IEEE, 2022, pp. 181–186.

[17] K. Ssonko, “Implementing device to device communication using software defined radios on 4g cellular systems,” Ph.D. dissertation, Makerere University, 2022.

[18] M. Vilakazi, C. R. Burger, L. Mboweni, L. Mamushiane, and A. A. Lysko, “Evaluating an evolving oai testbed: Overview of options, building tips, and current performance,” in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1. IEEE, 2021, pp. 818–827.

[19] F. John, J. Schuljak, L. B. Vosteen, B. Sievers, A. Hanemann, and H. Hellbrück, “A reference deployment of a minimal open-source private industry and campus 5g standalone (sa) system,” in *2022 IEEE 10th International Conference on Information, Communication and Networks (ICICN)*. IEEE, 2022, pp. 1–9.

[20] R. Poorzare and A. C. Augé, “Challenges on the way of implementing tcp over 5g networks,” *IEEE access*, vol. 8, pp. 176 393–176 415, 2020.

[21] “srsRAN performance,” 2023, Last accessed 9 April 2023. [Online]. Available: https://github.com/srsran/srsRAN_Project/tree/main/scripts/srsran_performance

[22] “Controlling processor C-state usage in Linux,” 2013, Last accessed 12 April 2023. [Online]. Available: https://docs.srsran.com/projects/project/en/latest/user_manuals/source/running.html

[23] “Building and installing the USRP open-source toolchain (UHD and GNU radio) on Linux,” 2022, Last accessed 01 February 2023. [Online]. Available: https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux

[24] “Running srsRAN project,” 2023, Last accessed 12 April 2023. [Online]. Available: https://wiki.bu.ost.ch/infoportal/_media/embedded_systems/ethercat/controlling_processor_c-state_usage_in_linux_v1.1_nov2013.pdf

[25] “Outputs-logs,” 2023, Last accessed 30 March 2023. [Online]. Available: https://docs.srsran.com/projects/project/en/latest/user_manuals/source/outputs.html

[26] “USRP time calibration,” 2023, Last accessed 05 March 2023. [Online]. Available: https://docs.srsran.com/projects/project/en/latest/user_manuals/source/troubleshooting.html

[27] “Building Open5GS from Sources,” 2022, Last accessed 05 February 2023. [Online]. Available: https://open5gs.org/open5gs/docs/guide/02-building-open5gs-from-sources/

[28] “Sysmocom,” 2021, Last accessed 08 February 2023. [Online]. Available: https://sysmocom.de/products/lab/sysmousim/index.html

[29] “Open-cells,” 2020, Last accessed 08 February 2023. [Online]. Available: https://open-cells.com/index.php/sim-cards/

[30] “PySim,” 2022, Last accessed 28 April 2023. [Online]. Available: https://github.com/osmocom/pysim

[31] “Pysim-suci,” 2022, Last accessed 20 March 2023. [Online]. Available: https://gist.github.com/mrlnc/01d6300f1904f154d969ff205136b753

[32] X.-b. Ye, “Design and performance study of 4g communication system based on srsran,” in *Second International Symposium on Computer Technology and Information Science (ISCTIS 2022)*, vol. 12474. SPIE, 2022, pp. 585–593.

[33] “5G COTS UE,” 2019, Last accessed 27 March 2023. [Online]. Available: https://docs.srsran.com/projects/4g/en/latest/app_notes/source/5g_sa_COTS/source/index.html

[34] “5G slicing,” 2023, Last accessed 28 April 2023. [Online]. Available: https://source.android.com/docs/core/connect/5g-slicing/

[35] “Android 13 showcase,” 2022, Last accessed 25 April 2023. [Online]. Available: https://news.abplive.com/technology/

[36] “TS 23.501,” 2022, Last accessed 12 March 2023. [Online]. Available: https://www.tech-invite.com/3m23/toc/tinv-3gpp-23-501_zi.html