Design and Analysis of an Open-Source SDN-based 5G Standalone Testbed

Diana Pineda*, Ricardo Harrilal-Parchment*, Kemal Akkaya*, Ahmed Ibrahim*, Alexander Perez-Pons*
*Dept. of Electrical and Computer Engineering, Florida International University, Miami, USA 33174
Email: {dpine033, rharr119, kakkaya, aibrahim, aperezpons}@fiu.edu

***Abstract*—The fifth generation of mobile wireless communication represents a significant evolution of mobile networks due to its promises of Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communications (URLLC), and Massive Machine Type Communications (mMTC) to be applied to real-world applications. Even though there have been 5G Network deployments in some parts of the world that depend on 4G/LTE Core Networks, 5G is still a work in progress, with ongoing research and development to improve and develop this technology. 5G testbeds are essential for ongoing research to propose a solution, simulate, configure, test, and evaluate the impact of different network parameters on the system. Therefore, this paper presents a comprehensive approach to deploying a Standalone (SA) SDN-based 5G testbed for researchers to study and improve the performance of 5G Networks. Our testbed includes integrating several open-source projects to provide two testing scenarios where we assess the throughput and latency performance, Network slicing configuration, and the feasibility of SDN capabilities. The testbed provides a helpful instrument and lessons learned for the research community to improve and contribute to the successful deployment of 5G Networks in the real world.**

***Index Terms*—5G testbed, SDN, SA, Open-source, Network Slicing**

I. Introduction

Since the introduction of the first-generation mobile network in the 1980s, mobile wireless communication requirements have been growing significantly from analog phone calls to real-world applications such as autonomous vehicles, remote surgeries, and advanced robotics. These real-world applications depend on three fundamental building blocks required for 5G Networks: Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communications (URLLC), and Massive Machine Type Communications (mMTC). These three fundamental building blocks are aimed at solving past generations' challenges, such as higher capacity, lower latency and cost, and consistent quality of service [1]. The fact that 5G will tackle these challenges is highly appealing to customers expecting to enhance their business or industry more efficiently.

To deliver these requirements, it is necessary to apply different technologies to provide customization, flexibility, and management in the 5G Network, such as Software-Defined Networks (SDN) and Network Function Virtualization (NFV) [2]. SDN and NFV will help to manage the resources better, enable scalability, manage the network from a centralized perspective, and restore system functionality after an attack. Also, it is possible to apply different mitigation services against anomalies in the network using machine learning.

Several researchers are working on proposing different solutions to speed up the process of 5G deployment toward achieving the 5G KPIs: eMBB, URLLC, and mMTC. Therefore, developing a testbed that simulates/emulates these functionalities as close as expected to real 5G Networks and demonstrating that these solutions will contribute to the overall 5G proliferation is crucial. However, building a 5G testbed can be challenging since it is necessary to balance the intellectual property and autonomy of the current implementations and the proposed new functionalities and features to optimize the technologies [3].

Currently, there are two main options in these research and development efforts: 1) purchase specialized-purpose hardware and software with 5G capabilities, or 2) utilize and deploy open-source tools and resources. The main advantage of using specialized-purpose equipment is that the deployment is straightforward. However, this is costly and hinders flexibility to deploy and test other approaches. On the other hand, open-source projects reduce costs and allow fine-grained customization. The challenge with this option is that the deployment process can become complex, and there is a lack of support. Still, most researchers will trend toward open-source interfaces due to cost perspectives and innovation opportunities.

This paper proposes a 5G testbed focusing on implementing Standalone (SA) mode in 5G Networks using SDN and NFV. We have utilized a range of open-source projects, including Open5Gs, srsRAN, UERANSIM, Opendaylight (ODL), and OpenvSwitch (OVS), to build and deploy our testbed, which is one of the first to provide a comprehensive understanding of how to deploy and configure 5G SA networks using SDN and NFV, focusing on the hardware and software components involved. In addition, detailed instructions on deploying and configuring these components are included for their easy reproducibility. Overall, our testbed represents a valuable resource for those interested in exploring the capabilities and potential of 5G SA networks using SDN and NFV with the exception of NFV Management and Orchestration (MANO) deployment.

Through this testbed, we conducted some preliminary experiments to demonstrate feasibility and performance. The testbed experiments were designed to measure the data throughput and latency, demonstrate Network Slicing (NS) capabilities, and how SDN can be utilized to manage the network traffic.

The rest of the paper is organized as follows. Section II provides related work to our testbed. Section III provides an understanding of the concepts mentioned in the paper. The proposed testbed approach is explained in section IV, followed by the experiments and results in Section V. Section VI wraps up the paper, while Section VII offers the details of the planned demonstration.

II. Related Work

The use of testbeds to evaluate the performance of 5G Networks has garnered significant interest in recent years, as they offer a controlled and reproducible environment for evaluating the capabilities and limitations of different 5G technologies and architectures. As a result, several previous studies have developed and deployed various 5G testbeds to evaluate the performance of different aspects of 5G Networks. In the previous study, [4] introduced an open-source 5G testbed with NS capabilities for research purposes. This testbed used Open Air Interface (OAI) Non-Standalone (NSA) elements and included a 5G SA element (Access and Mobility Management Function (AMF)) to allow interworking with Evolved-UMTS Terrestrial Radio Access Network (E-UTRAN). In addition, OAI was selected to simulate the User Equipment (UE) and Radio Access Network (RAN) since it allows adding the NS features for the registration procedure. Despite their great contribution of adding NS capabilities and providing details on how to replicate their approach, their testbed did not address SDN and 5G SA implementation.

Hence, [5] developed an open-source testbed with blockchain-enabled for Non-Public Network (NPN) architectures. The testbed's purpose is to test the solution's effectiveness for this architecture and other capabilities, such as exploring commercial applications. The authors' methodology is based on cloud and containerization to deploy free5GC as the 5G SA Core Network, UERANSIM as the UE and RAN simulation, and Go-Ethereum for the blockchain nodes. While this testbed implemented 5G SA network functions, it is an example of UERANSIM applications and focused mainly on how to replicate their blockchain approach. In addition, SDN and NS were not considered at all. Following this idea, [6] proposed an open-source 5G testbed that supports multi-tenancy, multi-radio access technologies, SDN functionalities, orchestrating capabilities, and the deployment of End-to-End (E2E) NS. This testbed implemented 5G NSA OAI for the Core Network, srsLTE for the RAN, Open Source Mano (OSM) as the NFV-MANO entity, and Openstack as the Virtual Infrastructure Manager (VIM). In addition to the integration of two tenant controllers: 5G-EmPOWER and M-CORD. Similar to many others, this testbed also did not implement 5G SA elements, and the usage of OpenStack limits the flexibility of customizing the SDN capabilities.

Moreover, in [7], the authors had a different approach by providing a portable demonstrator of their 5GENESIS project 5G experimentation. This approach aims to have a tool that allows on-site testing and experimentation. This portable demonstrator consists of several physical components, from commercial nodes to open-source projects: OAI for the 4G/5G RAN, Ettus Universal Software Radio Peripheral (USRP) N300 for RAN and UE simulation, Commercial Off-The-Self (COTS) 5G NSA UEs, and OAI to simulate the Core Network with 5G NSA components. The portable demonstrator displayed different components that can be used to deploy a 5G Network. Nonetheless, there are no 5G SA elements and NS approach implementation. Even though SDN was mentioned as a part of the components, there were no details about SDN setup and adequate details to replicate their approach. Also, in [8], the authors presented an open-source, scalable 5G security testbed for experiments and implementation of different security mechanisms that can be applied to real-world scenarios. The testbed components consist of Free5GC to deploy 5G SA Core Network functions, Openstack for Network Function Virtualization Infrastructure (NFVI) and VIM, OSM as the NFV-MANO, OpenFlow (OF) enabled switches, and Open Network Operating System (ONOS) as the SDN-controller. This method implemented several mechanisms for 5G security testbed experiments. However, the UE and RAN were not implemented, comprehensive manuals to replicate were not provided, and there was no demonstration of how SDN can manage the network traffic. Instead, other tools were implemented for that purpose.

Following this idea, authors in [9] considered rural areas and proposed a practical and scalable 5G testbed that manages and orchestrates network slices. The testbed combines open-source solutions: Open5Gs for the 5G SA Core Network, UERANSIM for the UE and gNB implementation, ONOS as the SDN controller, OpenvSwitches as OpenFlow enabled switches, Openstack as the VIM, and OSM as the NFV-MANO. The authors deployed two Core Network slices with an operator assigned for each slice. While this approach is closest to ours, it lacks details to replicate the testbed and demonstrate the SDN capabilities to manage the network.

III. Technical Background

A. 5G architectures

3GPP has created eight deployment versions of 5G technology, shown in Fig. 1, and introduced two classifications. Release 15 introduced Option 3, a NSA architecture that uses the existing 4G infrastructure. Subsequently, Release 16 introduced Option 2, a SA architecture that lets users appreciate the network's full potential by utilizing only 5G components. Along with these options, 3GPP introduced six other deployment options varying in the connectivity between each 4G and 5G element, as seen in Fig. 1.

B. UERANSIM

UERANSIM is an open-source project that simulates the 5G UE and gNB for SA and NSA architectures, supporting many simultaneous communications. UERANSIM is compatible with other open-source Core Network simulations such as Open5Gs, Free5GC, and others to set up a 5G testing environment [10].

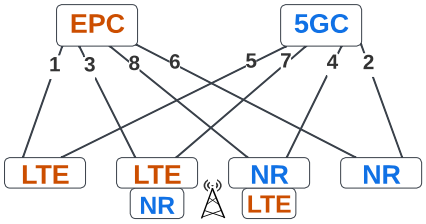

Fig. 1. 5G architecture

C. srsRAN

Software Radio Systems (SRS) is an Ireland-based company that builds open-source software for 4G and 5G mobile wireless software radio (UE and RAN) systems for commercial customization [11]. srsRAN is one of their software suites. srsRAN allows using other third-party Core Network solutions to build a complete 4G/5G mobile wireless network, and it is compatible with Software Defined Radios (SDR) such as USRPs, BladeRF, and others [12]. This software radio suite includes srsUE for a full-stack 5G NSA/SA application, srsENB for a full stack 5G NSA/SA capabilities, and srsEPC for a light-weight 4G EPC implementation [13].

D. Open5GS

Open5Gs is an open-source project written in C and allows portability for various hardware and application platforms. This project allows the building and configuring of 5G SA networks or 4G networks with 5G capabilities (NSA). It provides a scalable and flexible platform for deploying 5G Networks and services [14].

E. Software-Defined Networks

SDN refers to separating the network control plane from the forwarding plane [15]. The SDN paradigm is based on separating the control network functions from the network devices themselves, and they will become simple packet-forwarding devices while the control logic is implemented in the centralized SDN controller [16]. To achieve such separation, three components are required: a centralized SDN controller, SDN-capable switches, and a management protocol. SDN's architecture consists of the control plane, the data plane, and the application plane that communicates with the control plane [17].

F. Network Function Virtualization

NFV is the concept of moving network services from dedicated physical hardware/stand-alone appliances into services/software that runs in a virtualized environment, such as any white box or COTS server [18]. The NFV framework consists of three main components: 1) Virtual Network Functions (VNFs); 2) Network Function Virtualization Infrastructure (NFVI); and 3) NFV Management and Orchestration (NFV-MANO).

IV. Proposed 5G SA Testbed Approach

A. Early Considerations

During the early stages of developing a 5G testbed for research purposes, we conducted a thorough assessment of prior related work based on their infrastructure and used resources. We considered four different aspects of their testbed: CN, RAN, UE, VIM, and the SDN controller. After collecting the different open source projects used, we selected the most commonly used: OAI for the Core Network, UE and the RAN, Openstack as the VIM, and ONOS as the SDN controller.

1) Core Network, UE and RAN: We initially started with OAI. OAI is an open-source project that provides a 5G SA Core Network implementation and 5G RAN that allows a full 5G Network simulation. In addition, OAI offers several deployment modes (minimalist, basic, and slicing) controlled by a python script, uses docker containers for the NF and docker-compose to start, stop, and manage the 5G NF, and provides several tutorials and manuals for deployment. OAI was a great tool for our early research stages. However, the docker-compose deployment was not suitable for our approach since we were trying to split the network functions and deploy these services independently in different VMs since each network function relied on different parameters to deploy. OAI would be a better option for researchers who trend toward docker containers and are looking for portability and an isolated environment for the OAI software.

Regarding the RAN, OAI offers documentation explaining how to implement a basic deployment and test it with UERANSIM as a RAN emulator, resulting in a successful job. Since UERANSIM is flexible, customizable, and compatible with other open-source 5G Core Network implementations, we chose it to be integrated to our proposed testbed with Open5GS.

2) SDN: ONOS is an open-source SDN controller and provides a centralized control plane for network device management, such as switches and routers. During our initial stage, we deployed ONOS as our SDN controller in conjunction with Openvswitch for SDN testing. However, after working on the OpenStack deployment using DevStack, we exchanged the ONOS controller for Opendaylight due to much more available network plugins.

B. Final Choices

After evaluating and deploying several solutions for 5G Network simulation, SDN, and VIM, we finally encountered the best suitable tools for our testbed. Therefore, compared to the initial proposal, we changed our approach based on the following: **Open5GS** for the Core Network, **UERANSIM** for the RAN, **VMware ESXI** as the VIM and hypervisor, and **Opendaylight** as the SDN controller. The following sections will provide a better understanding of the final stage of building a 5G-SDN testbed with these choices.

1) VIM-NFVI: Compared to the initial stages, we selected VMWare ESXI as our VIM and Hypervisor which is a bare metal hypervisor. Using ESXi, it is possible to create multiple

VMs and configure the VM hardware easily. Also, one of the benefits of VMware ESXI is that it is possible to assign USB devices using the available USB controller. This feature was essential for us since, as an extension, we plan to deploy Ettus Software Defined Radio and RF drivers that communicate through USB 3.0.

2) Core Network: Open5GS is our final choice for the Core Network deployment. It provided the flexibility required for our testbed, which consisted of easy customization by modifying the configuration files of each NF. Also, it was compatible with other projects, such as UERANSIM and srsRAN, and allowed the CP and UP separation in a very convenient manner as opposed to OAI.

3) RAN and UE: Regarding the RAN, we considered two solutions: UERANSIM and srsRAN. UERANSIM provided a tool called NR-binder, which binds it with the Core Network and acts like N2, N3, and N4 interfaces. The advantage of this tool is that it allows different types of testing using the created tunneling interface (i.e., *uesimtun0*). In addition, several testing options were available, such as speedtest, ping, docker, curl, and python applications.

srsRAN, on the other hand, allows the UE and RAN simulation using a different approach via a messaging library called ZeroMQ. ZeroMQ is an asynchronous messaging library for distributed or concurrent applications to provide high performance. ZeroMQ ensures that the gNB can quickly and efficiently send and receive data [19]. Also, ZeroMQ allows the substitution of the need for physical RF hardware and transmit radio samples. Using srsRAN in conjunction with Open5Gs makes it possible to create an end-to-end network. This method requires deploying the UE and RAN on a single computer and separating them using different network namespaces. The UE will receive an IP address from the 5G Core, and the Linux kernel will bypass the tunneling interface during routing traffic between both ends.

4) SDN: An important and unique aspect of our testbed is to be able to use SDN capabilities. SDN will provide the network programmability that the 5G Network needs to keep agility and flexibility since it will deal with more traffic than other mobile wireless generation. As a result, using SDN will improve the network performance, facilitate adding new services, and reduce the complexity of managing the network [20]. There are several methods to deploy SDN since different SDN switches and controllers are available. After considering this, we kept this approach in mind to allow researchers to choose their suitable SDN controller with Openvswitch.

For our testbed, we used open-source projects to deploy the different SDN components. We use OpenvSwitch as our SDN-capable switches, Opendaylight as our SDN controller, and OpenFlow 1.3 as the management protocol. We selected OpenvSwitch as our virtual switch due to its flexibility and compatibility with several open-source SDN controllers. Also, it supports OpenFlow and can be used in various virtualized environments. When it comes to Opendaylight, as mentioned in the prior section, after getting familiar with it during the implementation in Openstack, we discovered the benefits of Opendaylight as an SDN controller. Opendaylight has multiple available applications that provide flexibility, several available documentation, and more straightforward deployment.

For the SDN components deployment, we created 2 VMs with the same configuration mentioned in the Core Network section: one for the OpenvSwitch, and one for the Opendaylight controller. All the VMs that contains the Core Network functions, base stations, and UEs are connected to the Openvswitch that the Opendaylight controller remotely controls. The motivation behind this strategy is to allow us to control all the component traffic of our testbed using the SDN controller. In a real-world scenario, the SDN switch can be placed in Midhaul or Backhaul, and the UEs may not be directly connected to the SDN switch. However, our approach is to allow other researchers to use it to better understand the behavior of the UE traffic within the 5G Network. It is also possible for other researchers to customize the proposed testbed and place the SDN switch in different segments of the network.

C. Different Testbed Scenarios

We implemented different scenarios using different tool options. In any case, we installed ESXI VMWare on top of a Dell Precision Workstation with 24 CPUs, 64GB of RAM, and a 1 TB SSD. We deployed the CP NF and UP NF in different virtual machines with 2 Core, 4G RAM and 40GB of Disk. Similarly, UERANSIM and srsRAN were also deloyed in two separate VMs. Finally, we created a VM for including a Gateway router at the boundary. The purpose of the gateway VM is to keep our testbed self-contained from the FIU network environment. The Gateway VM is directly connected to the ESXi virtual switch and the OpenvSwitch to allow the testbed components to access Internet. This Gateway VM serves as a boundary to isolate our 5G testbed from the host environment. The overall designs for these testbed scenarios are shown in Fig. 2, and 3. We now explain the details for each scenario. Comprehensive guides on how to build the proposed testbed can be found on: [1]

1) Scenario 1: Scenario 1 uses UERANSIM as the UE and RAN simulator with Open5Gs as the 5GCN, as shown in Fig. 2. In addition, two UE VMs has been created to run the UERANSIM script to initiate the UEs and create the corresponding UE tunnel interface. In this scenario, we also implemented the NS capabilities offered by Open5GS and UERANSIM. Using the Open5GS Web User Interface, it is possible to assign a network slice to a subscriber. Furthermore, it allows the user to assign a new slice by providing Slice/Service Type (SST), Slice Differentiator (SD), Slice priority, and QoS. To this end, we added an extra UPF (UP2) to the architecture. As a result, we set up two network slices and declared them into AMF, NSSF, SMF, UPF, and GNB configuration files. This means we needed to create an additional VM for UP on Fig. 2. During the subscriber registration, we assigned different slices to each UE. In addition to the network slice assignation, we also set

[1] https://github.com/adwise-fiu/5G-SDN-Testbed

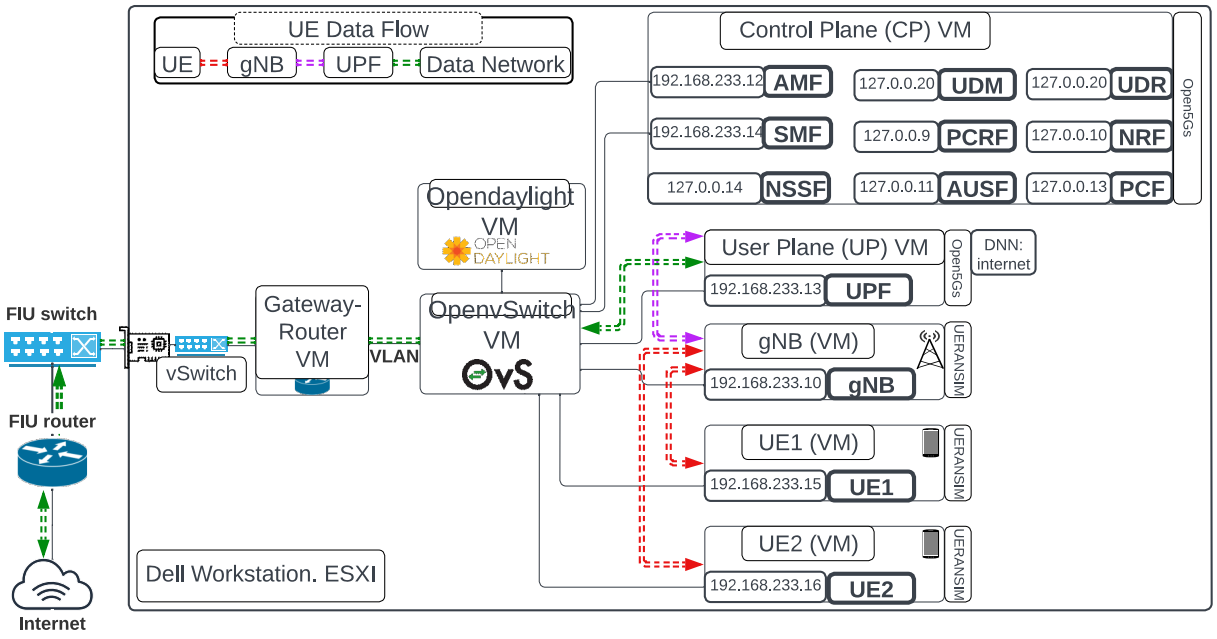

Fig. 2. Scenario 1 - Open5Gs and UERANSIM

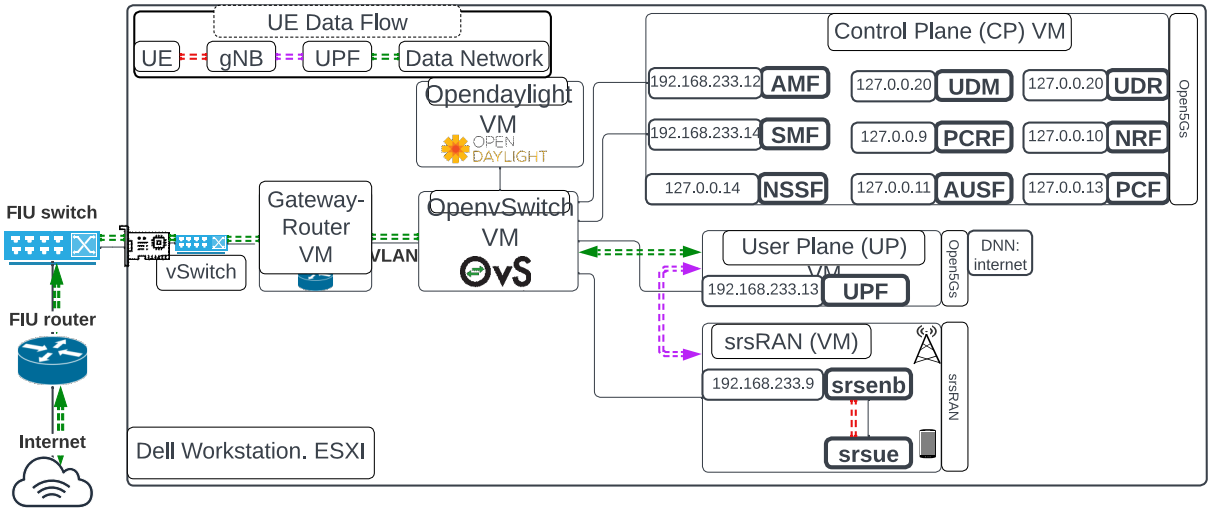

Fig. 3. Scenario 2 - Open5Gs and srsRAN

the Session Aggregate Maximum Bit Rate (AMBR) to control the amount of bandwidth a UE can use during a session. This feature ensures that the network's resources are used efficiently and that the UE does not consume more than its allocated share of network resources.

2) Scenario 2: Scenario 2 utilizes srsRAN for the UE and RAN simulator with Open5GS as the 5G Core Network. Compared to scenario 1, only a single gNB and UE is supported. Therefore, we decided to test the environment with one UE. The gNB and the UE are running in the same VM, as shown in Fig. 3.

V. Experiments and Results

Using the developed testbed, we conducted some preliminary experiments to evaluate the feasibility and performance of the proposed scenarios. The preliminary experiment involved deploying 5G equipment at a test site, collecting data on throughput, and latency from the UE simulations. To measure each scenario's bandwidth and latency, we used the speedtest-cli command-line interface tool for scenario 1 (Fig. 2) and scenario 2 (Fig. 3). This tool allows measuring the performance of a network connection from the UE perspective Speedtest CLI allows measuring the internet connection without relying on a web browser. The test consists of the Linux environment (the client) to determine the device location and the closest test server based on a ping response. Once the server has been established, the download test begins, which is the client trying to download a small piece of data from the server. During this test, two aspects are considered: the time it took the client to get a piece of data and how much network resources were used. If the client detects more sources available, it opens more

connections to the server and downloads more data [21]. The obtained results, which consisted of an average of ten samples taken every hour, are shown in Table I.

TABLE I
THROUGHPUT AND LATENCY

		Throughput Average (Mbps)	Latency Average (ms)
Scenario 1 - UERANSIM			
UE1	**DL**	98.27	15.08
	UL	111.0	
UE2	**DL**	112.82	13.73
	UL	131.68	
Scenario 1 - UERANSIM - NS			
UE1	**DL**	21.72	13.45
	UL	22.14	
UE2	**DL**	10.79	13.34
	UL	11.24	
Scenario 2 - srsRAN			
UE	**DL**	582.67	10.20
	UL	281.07	

As seen, scenario 1 presented a downlink data rate between 98.27 and 112.82 Mbps and an uplink rate between 111 and 131.68 Mbps. In contrast, scenario 2 showed almost a quintuple of the downlink average (582.67 Mbps) and a triple of the uplink average (281.07 Mbps) from scenario 1 results, as shown in Table. I. The reason behind these results and such a big gap between deployments using UERANSIM vs. using srsRAN is the ZeroMQ utilization. ZeroMQ provides high-performance results due to its asynchronous communication, multithreading, high-performance transport mechanisms, and high-throughput communication design [19].

On the other hand, checking the latency results, we observed that scenario 2 showed better results than scenario 1, with a latency of 10.20 ms, while that of scenario 1 was between 13.73 and 15.08ms as shown in Table. I.

VI. CONCLUSION

In this paper, we presented the design, development and analysis of a 5G SA testbed to offer an open-source tool for researchers while also evaluating how different UE-RAN simulators can influence the 5G Network performance and how SDN can be incorporated to control traffic. We incorporated software components, such as the 5G Core Network functions, gNB, UE, SDN switch, and SDN controller, to develop appropriate test scenarios and metrics for evaluating the testbed's performance. Our testbed provides a valuable tool for researchers looking to study and improve the performance of 5G Networks and can contribute to the successful deployment of 5G Networks in real life.

VII. DEMO

In this section, we explain what our demo will include in terms of setup and additional experiments. We plan to demonstrate NS and SDN capabilities through our demo.

A. Setup

The demo setup consists of using a laptop that will allow us to remotely access our setup with the mentioned Dell Workstation with VMWare ESXi as the hypervisor using SSH. Within this environment, we will have six VMs: one for the Control Plane (Open5Gs), two for different UPFs (Open5Gs), one for the gNB (UERANSIM), and two for the UE simulation (UERANSIM). The mentioned components are based on the Scenario 1 in Section IV.C. Using the mentioned equipment, we will run the scripts to start the UE and RAN simulation, as we are also displaying the interaction logs between these two projects. After the setup has been established, we will proceed to demonstrate the following experiments.

B. Network Slicing

For the NS demonstration using scenario 1 shown in Fig. 2, we assigned different network slices on each UE and verified that they had been reflected during the process of a user device connecting to the 5G Network. We will show the AMF logs, which provide the indicated Subscribed Network Slice Selection Assistance Information (S-NSSAI) after the UE has been successfully registered to the network. It is important to consider that this is a basic implementation of an NS scenario for studying and testing purposes.

```
AMF logs
[amf] INFO:InitialUEMessage
[amf] INFO:[Added] Number of gNB-UEs is now 1
[amf] INFO:RAN_UE_NGAP_ID[1] AMF_UE_NGAP_ID[3] TAC[1]
CellID[0x10]
[amf] INFO:[suci-0-001-01-0000-0-0-0000000001] known
UE by SUCI
[gmm] INFO:Registration request
[gmm] INFO:[suci-0-001-01-0000-0-0-0000000001]  SUCI
[amf] INFO:[imsi-001010000000001:1] Release SM context
[204]
[amf] INFO:[Removed] Number of AMF-Sessions is now 0
[gmm] INFO:[imsi-001010000000001] Registration complete
[amf] INFO:[imsi-001010000000001] Configuration update
command
...
[amf] INFO:[Added] Number of AMF-Sessions is now 1
[gmm] INFO:UE SUPI[imsi-001010000000001] DNN[internet]
S_NSSAI[SST:1 SD:0x1]
```

Fig. 4. AMF logs - UE1

Based on the obtained logs shown in Fig. 4 and 5, we will pay attention to the last line on each log displays the indicating UE's Subscription Permanent Identifier (SUPI), DNN, and Subscribed Network Slice Selection Assistance Information (S-NSSAI). This information matches the provided information during the subscriber registration using the Open5gs WebUI and confirms the support of NS assignment in our testbed.

We already conducted some experiments by setting the AMBR of UE to 20 Mbps for DL and UL throughput and 10 Mbps for the second UE. The results, shown in Table I, demonstrate the ability of the network to reduce the UE's bit rate depending on the assigned network slice. The results show that UE number 1 has a downlink data rate of 21.72 Mbps and an uplink of 22.14 Mbps, while the second UE showed a downlink average of 10.79 Mbps and an uplink average of 11.24 Mbps. The latency results shown in Table I are slightly better than that of Scenario 1 without NS, which indicates that there is even performance improvement with NS. We plan to highlight this during our demo.

```
AMF logs
[amf] INFO:InitialUEMessage
[amf] INFO:[Added] Number of gNB-UEs is now 2
[amf] INFO:RAN_UE_NGAP_ID[2] AMF_UE_NGAP_ID[4] TAC[1]
CellID[0x10]
[amf] INFO:[suci-0-001-01-0000-0-0-0000000002] known
UE by SUCI
[gmm] INFO:Registration request
[gmm] INFO:[suci-0-001-01-0000-0-0-0000000002]  SUCI
[amf] INFO:[imsi-001010000000002:1] Release SM context
[204]
[amf] INFO:[Removed] Number of AMF-Sessions is now 1
[gmm] INFO:[imsi-001010000000002] Registration complete
[amf] INFO:[imsi-001010000000002] Configuration update
command
...
[amf] INFO:[Added] Number of AMF-Sessions is now 2
[gmm] INFO:UE SUPI[imsi-001010000000002] DNN[mec]
S_NSSAI[SST:2 SD:0x1]
```

Fig. 5. AMF logs - UE2

C. SDN Capabilities

As the second component of the demo, we demonstrate the SDN features by creating two flows using the rest API of Opendaylight in Postman API tool to allow any traffic and the second one to drop traffic coming from the UEs. Postman allows to perform CRUD (Create, Read, Update, and Delete) operations using the Opendaylight REST API to edit current flows in the Openvswitch [22]. The experiment consists of adding the first flow using Postman to accept all traffic in the OpenvSwitch using the action *Normal* only. Then, we start scenario 1 and run the specified command in Fig. 6 to confirm that traffic is working normally.

```
speedtest-cli --source 10.45.0.3 --secure
Retrieving speedtest.net configuration...
Testing from Florida International University (131.94.186.114)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by Summit Broadband (Miami, FL) [13.49 km]: 8.887 ms
Testing download speed.....
Download: 146.16 Mbit/s
Testing upload speed.....
Upload: 113.56 Mbit/s
```

Fig. 6. Speedtest Results after first flow added

After we add the second flow that drops the GRPS Tunnel Protocol traffic which will block traffic coming from the UEs using scenario 1.

```
speedtest-cli --source 10.45.0.3 --secure
Retrieving speedtest.net configuration...
Cannot retrieve speedtest configuration
ERROR: <urlopen error timed out>
```

Fig. 7. Speedtest Results after second flow added

Fig. 7 displays the obtained output after running the same speedtest-cli command using the same interface. This output suggests that there was an issue connecting to the speedtest.net server due to that the packets coming from the tunneling interface (uesimtun0) has been dropped. This way we will be able to demonstrate the feasibility of SDN controller.

ACKNOWLEDGMENT

This research was funded by a National Centers of Academic Excellence in Cybersecurity grant (H98230-21-1-0324 (NCAE-C-002-2021)), which is part of the US National Security Agency and National Science Foundation under the grant #2147196.

REFERENCES

[1] S. Sullivan, A. Brighente, S. A. P. Kumar, and M. Conti, "5G Security Challenges and Solutions: A Review by OSI Layers," IEEE Access, vol. 9, pp. 116 294–116 314, 2021, conference Name: IEEE Access.

[2] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, M. Paolino, E. Sfakianakis, F. Liberal, and G. Xilouris, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN," Computer Standards & Interfaces, vol. 54, pp. 216–228, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548916302446

[3] "5G Testbed - IEEE Future Networks." [Online]. Available: https://futurenetworks.ieee.org/topics/5g-testbed

[4] A. Shorov, "5G Testbed Development for Network Slicing Evaluation," in 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Jan. 2019, pp. 39–44, iSSN: 2376-6565.

[5] H. Liu, Z. Liu, M. Zhao, and Z. Gao, "A Blockchain-based Containerized Mobile Communication Testbed on Open Cloud Platform," in 2022 IEEE International Conference on Communications Workshops (ICC Workshops), May 2022, pp. 1–6, iSSN: 2694-2941.

[6] A. Esmaeily, K. Kralevska, and D. Gligoroski, "A Cloud-based SDN/NFV Testbed for End-to-End Network Slicing in 4G/5G," in 2020 6th IEEE Conference on Network Softwarization (NetSoft), Jun. 2020, pp. 29–35.

[7] P. Matzakos, H. Koumaras, D. Tsolkas, M. Christopoulou, G. Xilouris, and F. Kaltenberger, "An open source 5G experimentation testbed," in 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), Sep. 2021, pp. 1–2.

[8] A. K. Murthy, R. Parthasarathi, and V. Vetriselvi, "Security Testbed for Next Generation Mobile Networks," in 2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP), Feb. 2020, pp. 122–129.

[9] B. Koné, A. D. Kora, and B. Niang, "Network Resource Management and Core Network Slice Implementation: A Testbed for Rural Connectivity," in 2022 45th International Conference on Telecommunications and Signal Processing (TSP), Jul. 2022, pp. 200–205.

[10] A. Güngör, "aligungr/UERANSIM," Dec. 2022, original-date: 2019-09-25T05:44:14Z. [Online]. Available: https://github.com/aligungr/UERANSIM

[11] "SRS | Homepage | The most trusted open software for mobile wireless networks," Apr. 2021. [Online]. Available: https://www.srs.io/

[12] R. Mihai, R. Craciunescu, A. Martian, F. Li, C. Patachia, and M. Vochin, "Open-Source Enabled Beyond 5G Private Mobile Networks: From Concept to Prototype," Nov. 2022.

[13] "srsRAN 22.10 Documentation — srsRAN 22.10 documentation." [Online]. Available: https://docs.srsran.com/en/latest/

[14] S. Lee, "Quickstart," Dec. 2022. [Online]. Available: https://open5gs.org/open5gs/docs/guide/01-quickstart/

[15] "Software-Defined Networking (SDN) Definition." [Online]. Available: https://opennetworking.org/sdn-definition/

[16] H. Kim and N. Feamster, "Improving network management with software defined networking," IEEE Communications Magazine, vol. 51, no. 2, pp. 114–119, Feb. 2013, conference Name: IEEE Communications Magazine.

[17] J. Rischke and H. Salah, "Chapter 6 - Software-defined networks," in Computing in Communication Networks, F. H. P. Fitzek, F. Granelli, and P. Seeling, Eds. Academic Press, Jan. 2020, pp. 107–118. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128204887000189

[18] D. Huang, A. Chowdhary, and S. Pisharody, : From Theory to Practice. Boca Raton: CRC Press, Dec. 2018.

[19] "ZeroMQ." [Online]. Available: https://zeromq.org/

[20] S. K. Routray and K. P. Sharmila, "Software defined networking for 5G," in 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Jan. 2017, pp. 1–5.

[21] "Speedtest CLI: Internet speed test for the command line." [Online]. Available: https://www.speedtest.net/apps/cli

[22] "Postman API Platform | Sign Up for Free." [Online]. Available: https://www.postman.com