Building an Open Source Containerized 5G SA Network through Docker and Kubernetes

Larisa-Mihaela Tufeanu*, Alexandru Martian*, Marius-Constantin Vochin*,
Constantin-Laurentiu Paraschiv†, and Frank Y. Li‡
*Department of Telecommunications, University Politehnica of Bucharest (UPB), 061071 Bucharest, Romania
†Dept. of Automa. Control & Ind. Informat., University Politehnica of Bucharest (UPB), 061071 Bucharest, Romania
‡Dept. of Information and Communication Technology, University of Agder (UiA), N-4898 Grimstad, Norway
Email: {larisa.tufeanu; alexandru.martian; marius.vochin; cparaschiv3112}@upb.ro; frank.li@uia.no

2022 25th International Symposium on Wireless Personal Multimedia Communications (WPMC) | 978-1-6654-7318-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/WPMC55625.2022.10014753

***Abstract*—Automating software orchestration and service development represents the newest trend in the development of fifth generation (5G) core network (CN) as it enables flexible and scalable service deployment. The building blocks for such a trend include *Containers, Docker, Kubernetes*, and other orchestration methods that facilitate easy scaling, management and control, load balancing, and personalized quality of service. In this paper, we develop a containerized 5G standalone (SA) network, building two types of network topologies for 5G SA deployment based on the concepts of 5G *cloud network functions, Docker containers* and *Linux virtualization*. Based on our implementation of both *Minimalist Deployment* and *Basic Deployment*, an assessment on the attach procedure is performed through next generation application protocol (NGAP) filtering along with subscriber information. Moreover, emulated transmission control protocol (TCP)/user datagram protocol (UDP) traffic is injected into the network and its performance is evaluated based on metrics such as traffic volume and data rate for both uplink and downlink.**

I. Introduction

The concepts of network function virtualization (NFV), software defined networking, and software development containerization have facilitated the transition of service provisioning from dedicated hardware based deployment to a flexible software enabled procedure. Such a transition leads to much shorter service development cycles and reduced business costs [1]. From the perspective of fifth generation (5G) service provisioning, various network components and nodes can be virtualized as part of the NFV infrastructure, running on commercially available servers. Considering the diversity of various services and vertical applications, the architectural implementations of 5G networks vary from component options/deployment types to operation system (OS)-level virtualization platforms, or/and orchestrator selection. These variations are enabled based on containerized network function (CNF) particularization. Furthermore, it is expected that container based techniques and methodologies performed in a cloud-native manner [2] bring significant benefits of virtualization and containerization to 5G applications, supporting better flexibility and scalability.

Recently, there is surge of research and development activities towards network softwarization, containerization, and cloudification for 5G and beyond 5G (B5G) networks [3] [4]. In line with various ongoing activities following this trend, it is of great interest to experiment how to build an open source based containerized 5G standalone (SA) network with various number of required network functions and components as well as to investigate the performance of such a network.

In this paper, we present an experimental implementation of containerized 5G core (5GC) network which supports two types of 5GC deployment, namely, *Minimalist Deployment* and *Basic Deployment*, each consisting of different number of CNFs involved. The objective of this paper is to first implement these two types of 5GC deployment and then evaluate the performance of them. The implementation is developed based on the OpenAir-CN-5G platform [5] and in our implementation each Linux Ubuntu container emulates a 5G CNF with individual functionalities within the same network subnet. Such a containerized solution ensures proper connectivity and ease of parameter configurations for 5G operators. To assess the performance of our implementation with respect to both session establishment and data transfer, emulated transmission control protocol (TCP)/user datagram protocol (UDP) user data traffic flows are injected into the experimental 5G network. With further enhancement through container orchestration by Kubernetes, it is expected that the number of CNFs could be adjusted (i.e., scaling up or down) automatically according to resource and service requirements.

The rest of the paper is structured as follows. We first give an introduction on 5GC network in Sec. II and then present the concepts of Docker and Kubernetes in Sec. III. Afterwards, an overview of a 5GC platform, OpenAir-CN-5G, as well as our implementation steps are given in Sec. IV. The experiments performed based on our implementations are presented in Sec. V. Finally, conclusions are drawn in Sec. VI.

II. Introduction to 5G Core Network

In this section, we give a brief introduction on 5GC with respect to its constituent components, network functions, and deployment modes.

A. 5G Network Components

5GC is designed based on the concept of service based architecture (SBA) and it provides flexibility for service development by defining multiple network components which

978-1-6654-7318-7/22/$31.00 ©2022 IEEE

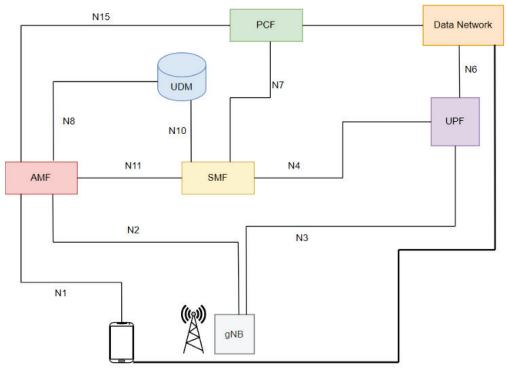


Fig. 1: An overview of the 5G core network architecture.

form the building blocks for service provisioning. One of the primary features of the core is ensuring the connection to the Internet and the public switched telephone network (PSTN).

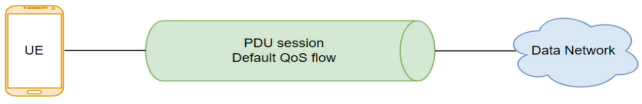
- **Access and mobility function (AMF)** handles processes describing user equipment (UE) mobility through the network, from authentication and authorization to security and signaling.
- **Session management function (SMF)** handles tasks including Internet protocol (IP) address allocation to UE attachment to the network, based on a set of parameters (e.g., international mobile subscriber identity (IMSI) and access point names (APNs)).
- **User plane function (UPF)** is the anchor point for packet data unit (PDU) session establishment. Its main feature is connecting user data from the radio access network (RAN) to the external data network (e.g., the Internet).
- **Unified data management (UDM)** acts like a database and ensures that operators are able to fetch user information, such as subscriber profiles, to permit or deny access to various services.
- **Policy control function (PCF)** is essential in implementing policy control within specific dynamic processes, based on particular time decisions.

As shown in Fig. 1, interconnections among these network components have been specified by the 3rd generation partnership project (3GPP) through reference points N (1-59) [6].

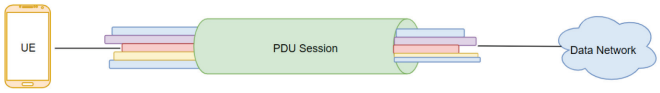
Furthermore, the control plane (CP) and user plane (UP) deal with signaling and session management among network components and user data traffic respectively. Accordingly, a PDU session could be established when a UE is active within the home or a visiting network. To ensure quality of service (QoS), various resource allocation mechanisms need to be adopted since Internet protocol (IP) services are initially based on *Best Effort*. In Fig. 2, we illustrate two basic cases for PDU session establishment between a UE and a data network with individual and multiple flows respectively, where QoS is achieved through default QoS flow or traffic prioritization.

B. 5GC Functions

Considering the availability of interconnections and network function components, the main 5GC roles are outlined below.



(a) A PDU session with default QoS flow.



(b) A PDU session with multiple flows.

Fig. 2: Illustration of single and multiple PDU sessions.

TABLE I: UP/CP Allocation in 5G NSA and 5G SA solutions

Technology	4G		5G	
	UP	CP	UP	CP
5G NSA		X	X	
5G SA			X	X

1) Routing user traffic – from source (the initiating UE) to destination (the external data network) and vice versa;
2) Providing QoS – Enforcing QoS policies, traffic prioritization, and resource reservation for specific services;
3) Ensuring security and confidentiality;
4) Supporting roaming functionalities from a tracking and management perspective;
5) Identification and service of visiting devices;
6) Ensuring voice services (e.g., voice over new radio (VoNR) and voice mail);
7) Providing accounting and billing services.

C. 5G Architecture: Non-Standalone versus Standalone

To ease the transition from fourth generation (4G) to 5G, two deployment architectures are specified by the 3GPP [7].

Non-standalone (NSA) deploys a complete end-to-end (E2E) 5G network by taking the advantage of virtualization processes, CP and UP separation (CUPS), and software defined networking (SDN). This architecture delivers 5G services with greater data rates than the ones obtained in 4G, by using an evolved packet core (EPC) core combined with a 5G new radio (NR) RAN.

Standalone. Unlike the NSA architecture, an NR RAN in the 5G SA solution is connected directly to the 5G core network. As such, SA supports the full set of 5G services.

Although 5G NSA has indeed an advantage in terms of faster deployment and a lower cost of implementation, it comes at the expense of network performance. On the other hand, 5G SA delivers full 5G benefits, such as ultra-high Internet access speed, and ultra-low latency. Another major difference between NSA and SA is CUPS, as in NSA networks UP is using the 5G stack and the CP is using the 4G stack, whereas in 5G SA networks the 5G stack is used for both UP and CP, as shown in Tab. I.

III. Containers, Docker, and Kubernetes

Containers, virtually seen as synonymous with cloud-native development as a process, represent software tools that provide more feasible packaging and isolation of applications within

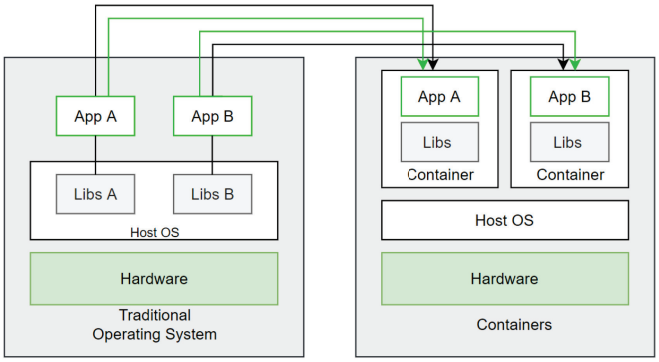


Fig. 3: Container versus standard OS differences [8].

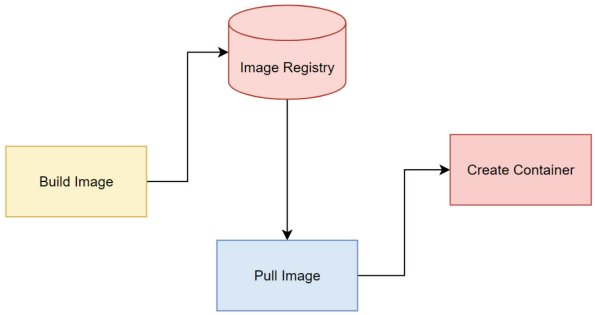


Fig. 4: Life cycle management of a container.

the runtime environment. Developing applications locally does not only involve hardware costs but also creates dependency issues in operations like migrations or reproduction. Consequently, emulators have gained popularity within the development stages, by enabling successful run of applications across various environments with minimal quality assurance degradation [8]. Considering migrations of applications, containers provide the advantage of bundling together all necessary files and dependences. The concept is very similar to a Linux distribution [9], where the Redhat package manager (RPM) packages and configures files. Although visual machines rely on virtualization at the hardware level, containers operate only at software layers above the OS level for resource allocation of central processing unit (CPU), random-access memory (RAM), disk space, etc. [8]. The main difference between container based and OS specific application development is shown in Fig. 3.

Furthermore, the life cycle management (LCM) tasks of a container involve various stages of development and they are rarely built from scratch. Inside LCM, public or private registries enabled operations are performed with container images in a database format, making more accessible their pulling, distribution, and storage.

With OS-level virtualization impact, *Docker* is one of the most popular *platform as a service (PaaS)* product suites. While containers introduce isolation, Docker not only enhances an economical resource allocation within a network but also packages applications and all stated dependencies into a container environment at a high scalability level, making them available from various locations and operating systems (Linux,

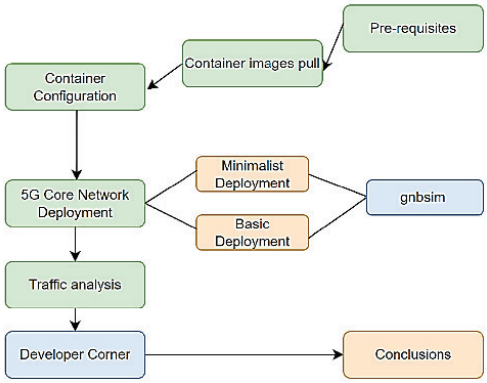


Fig. 5: 5GC network deployment implementation steps.

Windows, MacOS devices). Fig. 4 illustrates the application encapsulation procedure according to LCM.

Furthermore, Docker does not only contribute to the delivery of applications and services to end users (through the deployment and operation frame) but also provides a key component of the DevOps community. It takes part in a bigger automation picture by planning (Git, Jira), building (Gradle, Maven), testing (Selenium), and monitoring (Nagios).

On the other hand, *Kubernetes* is a strong representative of orchestration processes, running in container based applications. This tool brings capabilities of auto-scaling, life cycle management, declarative models, resilience and self-healing, persistent storage, and efficient distribution of workloads across clusters [8]. Although there is a common misconception that either Kubernetes or Docker should be used in a network infrastructure, they are indeed fundamentally distinct and complementary from each other for building enhanced containerized application capabilities [10].

IV. OpenAir-CN-5G Platform

OpenAir-CN-5G [5] is a 3GPP specification compliant 5GC implementation developed by the *OpenAirInterface (OAI) Software Alliance* which is lead by EURECOM and it provides a platform for our implementation and experiments.

A. Overview and Prerequisites

OpenAir-CN-5G is distributed under OAI Public License v1.1 containing a 5G SA architecture, and it is built on Linux containers using Docker. A Type-2 hypervisor for x86 virtualization and Oracle virtual machine (VM) VirtualBox was used for deploying a Linux Ubuntu 18.04 long term support (LTS) (Bionic Beaver) image. Fig. 5 offers an overview over the implementation steps required to implement a 5GC network.

The prerequisites for implementation include installing Linux open source networking packages and Docker installation & repository setup, allowing proper connections among apt hypertext transfer protocol (HTTP) components and adding Dockers gnu privacy guard (GPG) key. Creating a Docker Hub account is mandatory for command line interface (CLI) login and image pull activities, directly onto the Linux VM. Furthermore, a Linux Ubuntu distribution will be used as a container based image and MySQL database instances will

TABLE II: Deployment types and functions for the 5G SA network

Function	Minimalist Deployment	Basic Deployment
AMF	X	X
UDR		X
SMF	X	X
UPF	X	X
MySQL	X	X
NRF	X	X
AUSF		X
UDM		X

store subscription information required for the authentication and authorization processes. Based on the recommendation from [12], packet forwarding should be enabled to facilitate connectivity between *Docker Containers* and an *external data network (EDN)*, which is the Internet. To enable forwarding, Linux Kernel needs to be configured to allow IP forwarding by modifying the policy for *iptables* from *DROP* to *ACCEPT*. The purpose is to create a fully functional emulated 5G network instance, using specific CNFs as the core elements based on Docker images configured corresponding to each of the required functionalities. The images are hosted under an OAI account: oaisoftwarealliance [13]. To pull them from the Docker Hub, a valid user account is required. The images are experimental and refer to a 5G SA architecture.

After the images are pulled, they will be re-tagged to match the configuration scripts. As recommended by [13], the set of configuration files was initially pulled from (Gitlab) cn5g/oai-cn5g repository and then other necessary configuration data, such as the public land mobile network (PLMN) code, tracking area code (TAC), mobile country code (MCC), mobile network code (MNC), are appended according to each implementation.

B. Deployment Types

The scripts support the implementation of two different deployment types as explained below [13]. The functions that are supported by each deployment type are listed in Tab. II.

- **Minimalist Deployment** consists of the minimum number of network functions that are required to properly sustain the 5G infrastructure.
- **Basic Deployment** contains three additional CNFs – with authentication and authorization functionalities.

The authentication server function (AUSF) seeks to offer UE authentication assistance. AUSF is in charge of making decisions on authentication and it is also connected to a service for calculating authentication data (keys, authentication algorithms, such as 5G-AKA or EAP-AKA') [14]. The AUSF, UDM, and UDR functions comprise the home subscriber server (HSS) from 4G EPC networks (for stateless architectures). Therein, the UDM represents a network function that is native to the cloud and it stores, maintains, and controls network user data. Data for various use cases, such as subscription, policies, applications, and structured data, are stored in the UDR.

C. Gnbsim Simulator for RAN and UE

Gnbsim is an open source 5G SA simulator [15] compliant with the 3GPP Rel-16 standards and it is used for testing 5G

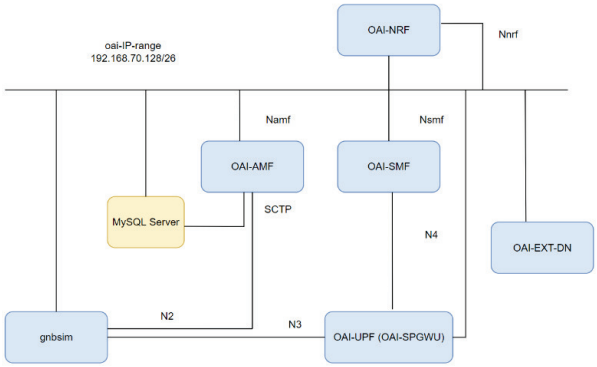


Fig. 6: Connectivity in the Minimalist Deployment architecture.

systems. *It simulates RAN and UE with one gNB and one UE per instance (per container).*

The gnbsim image is fetched from the oai-cn5g-fed workspace [16], and the Docker service is launched. Status commands should display up and healthy containers for both core and radio network deployment. Furthermore, initial results point to a successful UE attach to the network and a correct IPv4 allocation from the SMF. The subscriber information is fetched from the MySQL database and the parameters match with the ones configured on *Core*. A match permits a user to access the network, while a contradiction results in an access denial.

V. Implementations and Experimental Results

Based on the OpenAir-CN-5G platform, we have built a 5G SA network with both 5GC deployment types included. To form an E2E network, the gnbsim based container which functions as the RAN (including one gNB and one UE) is concatenated to the 5GC. As an example, the connectivity among the main components in the Minimalist Deployment architecture is shown in Fig. 6. In what follows, we present an overview about our implementation and the experiments.

A. Overview of Implementation and Experiments

Based on the implementation steps introduced in Fig. 5, we have performed 5GC deployment installations, supporting both container configuration and network prerequisites according to the deployment type selected. Consequently, an experimental 5G SA network is built under the oai-IP-range 192.168.70.128/26 and it represents a containerized solution ensuring ease of parameter configuration from the perspective a mobile network operator (MNO).

To assess the performance of the implemented network, we adopt *Wireshark* for protocol handshake validation and *Iperf* for data traffic performance evaluation. Specifically, the UE attached to the network allows *Ping* tests based on the Internet control message protocol (ICMP) for protocol handshake and traffic monitoring. A subscriber can not only ping the EDN but also carry user traffic. Domain name system (DNS) services are also functional as the UE has obtained an IPv4 address, 12.1.1.2, that is statically allocated by the SMF

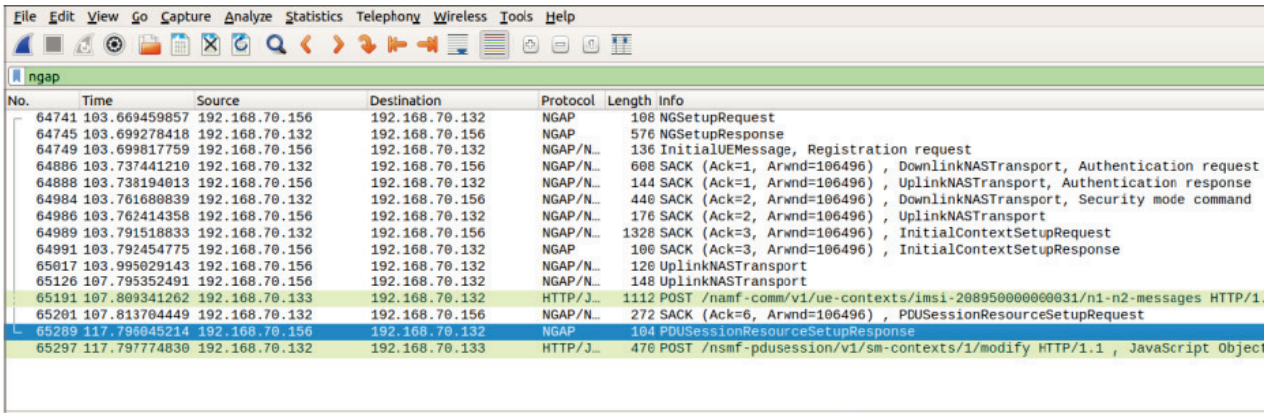


Fig. 7: NGAP messages in a 5GC Basic Deployment trace where authentication and security are required.

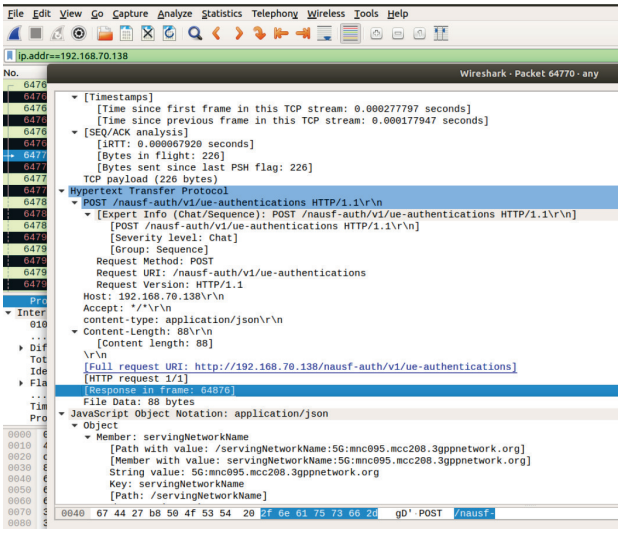


Fig. 8: AUSF filtering in a 5GC Basic Deployment trace.

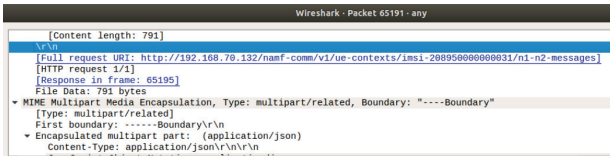


Fig. 9: IMSI filtering in a 5GC Basic Deployment trace.

during the PDU session establishment procedure. Through APN configuration, it ensures packet forwarding within the UPF or on the Intranet. Interconnections under the same subnet support ICMP operations (Ping) between the UE and the EDN (i.e., Google Server) based on the SMF configuration.

B. Protocol Handshake and Session Establishment

To filter the attach messages, the next generation application protocol (NGAP) filter needs to be applied in the captured trace. NGAP refers to the messages captured on the N2 reference point, which lies between the gNB and the AMF, and it provides control plane signaling information. As shown in Fig.7, the following attach messages are deducted:

1) A UE registration request is sent to the AMF (Message No. 64749);
2) The authentication and security process resulted from the Authentication request and Authentication response (Messages No. 64886-64888);
3) The PDU session establishment request describes the final Attach message (Message No. 65298).

Other filters such as HTTP or GPRS tunnel protocol (GTP) may also be used to filter other types of messages.

In Fig. 8, we illustrate the POST-type message referring to the AUSF-authentication HTTP request (to the complete request URI: http://192.168.70.138/nausf-auth/v1/ue-authentications) contained in this procedure. Additionally, the message includes information on the *Serving Network* name: mnc095.mcc.208.3gppnetwork.org, as well as the string values MNC=095, MCC=208. The result validates the successful match between the configured parameters delivered to the network through the UE request and the configuration files of the CNFs based on the MySQL database information. Furthermore, the ICMP packets related to the Ping queries from CLI are also visible in the Wireshark trace through request and reply messages.

C. Data Traffic Performance

Using Iperf, TCP/UDP data streams are generated for measuring network performance in terms of throughput and data rate. Accordingly, we start one Iperf process running in the server mode as the traffic receiver and then create another Iperf process running in the client mode on another host as the traffic sender. The sender and receiver are the gnbsim container (which represents the UE and gNB) and the external data network container (a Linux Ubuntu container with Internet connection) respectively.

The experiments are performed consecutively in 7 seconds and the throughput and data rate results are measured in an interval of every second. The measured results are illustrated in Tabs. III and IV respectively. To understand these values, let us take one example from the first interval, 0.00–1.00, of Tab. III. When an amount of 30.2 MB user data is transferred in 1 sec, the obtained throughput is $30.2 \times 8/1 = 241.6$ Mbps and this value is lower than the reported data rate, 254 Mbps, which is shown in the same row. This is because protocol overhead has been added when calculating data rates. The

TABLE III: Data transfer performance: 5GC Minimalist Deployment

External Data Network: oai-ext-dn			**Gnbsim**	
Interval (sec)	Transfer (MB)	Data rate (Mb/s)	Transfer (MB)	Data rate (Mb/s)
0.00-1.00	30.2	254	33.5	281
1.00-2.00	32.2	270	31.7	266
2.00-3.00	28.8	242	28.9	241
3.00-4.00	33.8	283	33.8	285
4.00-5.00	20.8	174	20.9	175
5.00-6.00	27.3	229	27.8	233
6.00-7.00	34.2	287	33.8	284

same interpretation applies to the other results shown in this table and similar results have been obtained for the second deployment type (illustrated in Tab. IV).

Finally, we present in Tab. V the average uplink and downlink data rates for each deployment type (either Minimalist Deployment which includes AMF, SMF, UPF, NRF or Basic Deployment which includes AMF, SMF, UPF, NRF, UDR, AUSFM and UDM). The values are obtained based on the results shown in Tabs. III and IV, and they are fairly appropriate (e.g., 30.05 MB versus 29.61 MB for the Minimalist Deployment, and 22.98 MB versus 22.65 MB for the Basic Deployment).

It is worth highlighting that the values obtained from the Minimalist Deployment are greater than those found in the Basic Deployment. More specifically, the data rate is approximately 60 Mbps higher in the Minimalist Deployment and the mean transfer volume per sec is approximately 7 MB larger. This is due to the fact that the Basic Deployment requires more signaling messages to be exchanged during the registration and authentication procedures caused by the additional cloud based network functions (AUSF, UDM, UDR).

VI. Conclusions and Future Work

In this paper, we reported an open source based containerized 5G SA network implementation which encompasses both Minimalist and Basic Deployment, and is compliant with 3GPP Rel-16 specifications. Various core network components are implemented based on the OAI 5GC platform and deployed using Docker-compose. Based on the implementation, both protocol handshake (including attach and session establishment) and data transfer (including TCP/UDP traffic flows) are demonstrated. The network performance based on the two deployment types is compared with each other through an E2E 5G network formed by emulating RAN and UE based on gnbsim. Our implementations and experiments demonstrate the significance of orchestration (through Kubernetes layering) as network size increases and how automation can contribute to enhancing 5G SA functionalities according to CNF components and services to be implemented. As our future work, we will explore Kubernetes at an orchestration level within the existing 5G infrastructure and trigger operations such as scaling and processes automation.

Acknowledgment

The research leading to these results has received funding from the NO Grants 2014-2021, under project contract no. 42/2021, RO-NO-2019-0499 - "A Massive MIMO Enabled IoT Platform with Networking Slicing for Beyond 5G IoV/V2X and Maritime Services (SOLID-B5G)".

TABLE IV: Data transfer performance: 5GC Basic Deployment

External Data Network: oai-ext-dn			**Gnbsim**	
Interval (sec)	Transfer (MB)	Data rate (Mb/s)	Transfer (MB)	Data rate (Mb/s)
0.00-1.00	23.8	199	25.5	213
1.00-2.00	13.0	109	12.9	109
2.00-3.00	20.7	172	21.4	180
3.00-4.00	32.2	272	32.7	275
4.00-5.00	27.9	234	28.2	233
5.00-6.00	22.5	189	21.2	180
6.00-7.00	18.5	155	18.4	154

TABLE V: Mean values for uplink and downlink in 5GC

Scenario	Minimalist Deployment		Basic Deployment	
	Transfer mean (MB)	Data rate mean (Mbps)	Transfer mean (MB)	Data rate mean (Mbps)
Uplink gnbsim	30.05	252.14	22.98	192.00
Downlink Data Network	29.61	248.42	22.65	190.00

References

[1] D. E. Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 256–281, 1st Quart., 2021.

[2] A. Pino, P. Khodashenas, X. Hesselbach, E. Coronado, and S. Siddiqui, "Validation and benchmarking of CNFs in OSM for pure cloud native applications in 5G and beyond," in *Proc. IEEE ICCCN*, Jul. 2021, pp. 1-9.

[3] R. Botez, J. Costa-Requena, I.-A. Ivanciu, V. Strautiu, and V. Dobrota, "SDN-based network slicing mechanism for a scalable 4G/5G core network: A Kubernetes approach," *Sensors*, vol. 21, no. 11, pp. 1–26, May 2021.

[4] N. K. Ostinelli, S. T. Arzo, F. Granelli, and M. Devetsikiotis, "Emulation of LTE/5G over a lightweight open-platform: Re-configuration delay analysis," in *Proc. IEEE GLOBECOM*, Dec. 2021, pp. 1–6.

[5] OAI, "Federation of the OpenAir CN 5G repositories," [Online]. Available: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/tree/master.

[6] D. Chandramouli, R. Liebhart, and J. Pirskanen (Eds.), "Appendix of 3GPP reference points," in *5G for the Connected World*, New York, NY, USA: Wiley, 2019.

[7] 3GPP, "System architecture for the 5G System (5GS)," TS 23.501, R17, v17.5.0, Jun. 2022.

[8] T. Larisa-Mihaela, "The impact of Kubernetes in 5G networks," *Master's Thesis*, University Politehnica of Bucharest (UPB), Romania, 2022.

[9] Redhat.com, "What's a Linux container?" 2019. [Online]. Available: https://www.redhat.com/en/topics/containers/whats-a-linux-container.

[10] Microsoft, "Kubernetes vs Docker — Microsoft Azure," 2022. [Online]. Available: https://azure.microsoft.com/en-us/topic/kubernetes-vs-docker/.

[11] 3GPP, "Release 15 description; Summary of Rel-15 work items," TR 21.915, R15, v15.0.0, Sep. 2019.

[12] Docker Documentation, "Use bridge networks," 2022. [Online]. Available: https://docs.docker.com/network/bridge/#enable-forwarding-from-docker-containers-to-the-outside-world.

[13] Gitlab, "OpenAirInterface 5G core network deployment: Pulling container images," 2022. [Online]. Available: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/RETRIEVE_OFFICIAL_IMAGES.md.

[14] CableLabs, "A comparative introduction to 4G and 5G authentication," 2019. [Online]. Available: https://www.cablelabs.com/insights/a-comparative-introduction-to-4g-and-5g-authentication.

[15] Eurocom, "Gnbsim," 2021. [Online]. Available: https://gitlab.eurecom.fr/kharade/gnbsim.

[16] Docker.com, "DockerHub," 2020. [Online]. Available: https://hub.docker.com/.