# Towards Stress Testing A UPF Node On A 5G Standalone Testbed

**Conference Paper** · September 2023

DOI: 10.1109/AFRICON55910.2023.10293284

CITATIONS
0

READS
448

**7 authors**, including:

Lusani Mamushiane
CSIR
**21** PUBLICATIONS **200** CITATIONS
SEE PROFILE

Albert A Lysko
Council for Scientific and Industrial Research, South Africa
**120** PUBLICATIONS **587** CITATIONS
SEE PROFILE

Joyce Mwangama
University of Cape Town
**65** PUBLICATIONS **259** CITATIONS
SEE PROFILE

Hlabishi Kobo
Council for Scientific and Industrial Research, South Africa
**23** PUBLICATIONS **656** CITATIONS
SEE PROFILE

# Towards Stress Testing A UPF Node On A 5G Standalone Testbed

Lusani Mamushiane[*,†], Albert A. Lysko[*,†], Thoriso Makhosa[†], Joyce Mwangama[*], Hlabishi Kobo[†]
[†]*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*—The global 5G technology market has been growing and is forecasted to continue at an accelerated rate in the foreseeable future, with over 225 million 5G smartphones sold globally in 2020. It is thus compelling for network operators to start investigating and assessing the readiness of existing and planned 5G network stacks, in order to deliver and continuously meet the envisaged proliferation of 5G subscribers. The core network, particularly the User Plane Function (UPF), dominates the performance of a 5G mobile network. This makes it critical for both network equipment vendors and operators to evaluate the performance of their UPF ecosystem. This paper takes an initial step towards UPF performance profiling using stress testing procedures. The measured performance metric is the CPU utilisation by a UPF Network Function under network traffic strain. We leverage open-source 3GPP-compliant 5G New Radio stack by UERANSIM and a Core Network implementation by Open5GS projects. Our results show that the Open5Gs UPF implementation can achieve up to 200 concurrent data connections using only 17% of the virtual machine's CPUs and 200 control traffic connections using only 15% of the same CPUs.**

***Index Terms*—5G Standalone, Stress testing, Load testing, User Plane Function, UPF, Core Networks, Control traffic, User traffic.**

## I. Introduction

With the steadily growing number of devices and applications that rely on access to the Internet, the demand for data has been growing exponentially [1]. To accommodate this growth and to allow for more flexibility in services, the widely spread fourth-generation (4G) of mobile communication networks, i.e. Long-Term Evolution (LTE), is being replaced with the fifth-generation (5G) communication systems. Supporting 5G requires new, more advanced, complex and expensive equipment on both operator's and user's sides.

Hence, the uptake of fifth-generation (5G) mobile networks has been relatively slow, especially in the greater African context. Most deployments have been based on the intermediate non-standalone (NSA) architecture. This is because the NSA architecture allows reusing and mixing 4G equipment with 5G components thus enabling a gradual transition with lower immediate costs. In South Africa, mobile network operators (MNOs) such as MTN, Rain, and Vodacom are among the few mobile network operators that have launched 5G NSA networks using the emergency temporary spectrum made available by the Independent Communications Authority of South Africa (ICASA) during the COVID-19 pandemic in 2020, to respond to the surge in network data demand. The main driver towards NSA-focused 5G rollouts is that network operators want to continue making money from their existing network assets and support existing 4G customers, and so adopt 5G gradually (as the number of users with 5G handsets grows), and collect revenues today.

The full capabilities of 5G are brought about by a standalone (SA) architecture. On a global scale, about 20 operators in 16 countries have launched 5G SA networks, since the first launches in China in 2020 (China Mobile, China Telecom, and China Unicom), South Africa (Rain) and the US (T-Mobile). According to a GSMA forecast [2], there will be more than 340 million 5G connections in Africa by the end of 2030, which is equivalent to a fifth of total mobile connections. Furthermore, the ecosystem of 5G standalone (SA) devices has also seen a noticeable growth, from 278 in 2020 to 663 in 2022 – a 138% growth [3]. This shows that 5G will be the biggest growth market in the foreseeable future. Network operators need to go beyond the launch of 5G to scale up in order to meet the envisaged surge in the 5G subscriber base.

Therefore, it is important to start to investigate and assess the readiness of existing and planned 5G network stacks to deliver and continuously meet the performance demands of their customers. This assessment requires a comprehensive load/stress test of the different elements of the 5G network, according to specific traffic and consumption profiles. Among these network elements is the Core Network (CN), which is considered the brain of any cellular network. The CN is mainly responsible for signaling and connecting actual data that pass through the Radio Access Network (RAN) to the Internet.

In 5G networks, the User Plane Function (UPF) — which is roughly equivalent to the 4G Serving Gateway (SGW) and Packet Gateway (PGW) in LTE/4G — is the main component of the CN responsible for carrying data and passing on network usage information. The UPF plays a pivotal role in the 5G network to realise the promise of low latency and high throughput [4]. Therefore, in order to measure the performance of the CN, the UPF needs to be stress tested to quantify how quickly and accurately it routes data packets, which is key to improving resource efficiency and user satisfaction.

There exist several free and open-source projects that focus on 3GPP-compliant 5G CN (bundled with UPF) implementations. Table I lists and compares some of these

TABLE I: Comparison of projects focused on 3GPP-compliant UPF implementations

| **Project** | **Open Source?** | **3GPP Compliance** | **Production Grade** | **Deployment Complexity** | **Cloud-native?** | **Licence** | **Number of Contributors** |
|---|---|---|---|---|---|---|---|
| **Magma [5]** | Yes | Release 16 | Academic product | Easy to deploy and use | No | Apache-2.0 | 10 |
| **Open5Gs [6]** | Yes | Release 17 | Semi-academic product | Fast and easy to deploy | Yes | AGPL license | 66 |
| **Free5GC [7]** | Yes | Release 15 | Semi-academic product | Easy to deploy and use | Yes | Apache-2.0 | 30 |
| **OAI 5GC[8]** | Yes | Release 16 | Academic product | Easy to deploy, difficult to use | Yes | OAI Public License v1.1 | Not clear |
| **OMEC [9]** | Yes | Release 16 | Semi-academic product | Difficult to deploy and use | Yes | Apache-2.0 | 24 |
| **Open5GCore [10]** | Yes | Release 16 | Semi-academic product, | Easy to deploy, difficult to use | Yes | Paid licensing model | Not disclosed |
| **Cumucore [11]** | No | Release 16 | Commercial product | Easy to deploy and use | Yes | Paid licensing model | Not disclosed |

notable projects and a few proprietary products. A CN that meets the following criteria is typically preferred by a majority of network operators:

- **Open source design** [12] — to minimise/avoid vendor lock-in and enable the introduction of new features to meet current customer needs on the fly.
- **Support for the latest 3GPP technical specifications** — to offer competitive services to customers.
- **High-production readiness**[13] — the product should not be too academic and should require minimal effort to get ready for production. This can be determined based on the main sponsors of product development, the stability of community support, and product trials to date.
- **Deployment complexity** [14] — the product should be easy to deploy and use for operators to accelerate capability building, reduce time-to-market and save costs on skills development.
- **Cloud-native design** [15] — to enable the containerisation of the CN on virtual machines or bare metal depending on the operator's use case.

This paper summarises our first step towards UPF performance profiling through stress testing. We opted for the Open5Gs CN implementation for our tests, largely because of its adoption coverage, wide community support, and support for the latest 3GPP specifications, compared to its rivals. There are limited research efforts geared toward profiling and optimising the performance of UPF implementations. To the best of our knowledge, there exist a total of three (3) related studies in literature, all unpacked in Section I-A.

## A. *Related Works*

Chen et al.[16] propose a UPF implementation optimised to run on a Docker containerised computing environment to provide a more flexible and scalable deployment. The authors utilised the Intel Data Plane Development Kit (DPDK) [17] to accelerate packet processing and Pktgen to model large volumes of traffic. Based on their stress tests, the authors were able to show that their UPF implementation can deliver up to 40 Gbps in throughput, and that the optimal throughput reaches about 60% with 64-byte packet size and 100% throughput with 256-byte packet size when running on a server with two CPU cores.

Zhou et al. [18] proposed and implemented a framework they dubbed "Cable" to accelerate UPF packet processing and optimise UPF CPU usage efficiency. Cable achieves this by scheduling PDU sessions across multiple UPFs (ensuring load balancing) and taking advantage of Linux kernel technologies, such as the extended BSD Packet Filter (eBPF) [19] engine and the eXpress Data Path (XDP) [20]. The authors used the Free5GC core network as a case study for their implementation and were able to demonstrate that Cable can reduce packet processing time in a UPF by at least 30% and improve UPF usage by 25%.

Thiago et al. [21] propose an in-kernel 3GPP Release 16-compliant 5G UPF optimised for deployment in a resource-constrained environment like a Multi-Access Edge Computing (MEC) [22]. In this context, the UPF and RAN are located together on the same MEC host, allowing for optimised performance and streamlined communication. Similar to Zhou et al. [18], the proposed UPF solution leverages the eBPF/XDP for fast packet processing. The authors were able to show that their proposed solution can scale and achieve a packet rate of 10 Mpps using only 60% of the CPU with 6 cores.

## B. *Contribution*

This paper intends to provide some initial 5G CN performance stress test plans and results — especially focussing on the UPF CPU and memory (RAM) usage efficiency, both translating to UPF node responsiveness. As can be seen in Section I-A, related studies have focused mainly on optimising the performance (particularly CPU throughput optimisation) of UPF nodes, taking advantage of Linux hardware acceleration technologies. The results and frameworks presented by these works are invaluable as they provide a basis for optimising resource utilisation in 5G. Notably, these works did not feature a full-fledged 5G mobile network setup in their optimisation experiments. Instead, their testbeds only constituted a standalone UPF connected to a traffic generator. We view this as a shortcoming, as it does not fully represent a real 5G network deployment scenario. To close this gap, we propose a test method for stress testing a UPF node running as part of a fully-fledged 3GPP-compliant 5G mobile network. Not only does our work provide insights on UPF CPU utilisation efficiency, but we also quantify the CPU utilisation of a UPF as a result of both data/user plane traffic and signalling/control traffic.

The key beneficiaries of the 5G UPF stress testing are:

- **Mobile Network Operators** — to ensure that their UPF nodes can deliver and meet performance demands for all services by validating vendor promises regarding scale and performance.
- **Researchers and Developers** — in order to integrate and test new applications and features on a 5G network, researchers and developers should have visibility into the UPF performance to ensure alignment of their solutions with the network capabilities.
- **Network Vendors** — manufacturers of network elements (such as UPFs) require stress testing procedures to prove the performance of their products, which also influences product pricing.

### C. *Organisation of Paper*

The paper is organised as follows: Section II provides an overview of the key building blocks of the experimental setup and their configurations. Section IV describes the test procedure followed for the stress tests as well as some implementation challenges encountered. Section IV presents and discusses the results. Finally, Section V concludes the paper and provides a direction for future research.

## II. Experimental Setup /Testbed Overview

The testbed setup is as depicted in Figure 1.

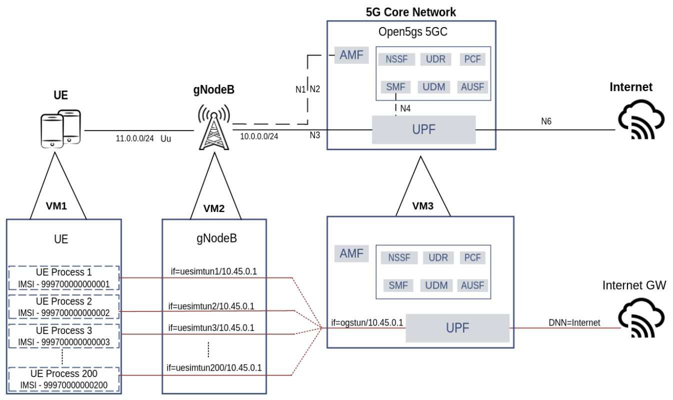

Fig. 1: Experimental setup

The setup broadly consists of four (4) major components: (i) the User Equipment (UE) domain, (ii) the Radio Access Network (RAN)/gNodeB domain, (iii) the Core Network (CN) domain, and (iv) the Internet. All of these components (except for the Internet) are running on Virtual Machines (VMs) deployed on OpenStack — an open-source cloud computing environment. The networking between the VMs is controlled by Neutron [23] — an OpenStack networking service responsible for providing network connectivity as a service to VMs. The deployment details of each of the components are provided in the subsequent sections.

### A. *User Equipment (UE)*

To profile the UPF performance, an emulated UE stack was deployed, based on UERANSIM [24] emulator. This UE stack can be considered an emulated 5G mobile phone implementation equipped with a virtual SIM card. This choice was made to target a test scenario which mimics a real production network environment. We created many emulated handsets, similar to what one would find in operational networks.

Notably, UERANSIM does not implement radio protocols below the RRC layer of the 5G New Radio protocol stack (see [25] for an overview of the 5G radio protocol stack). Therefore, the PHY, MAC, RLC, and PDCP are not implemented. The radio interface is partially simulated over the UDP protocol. This limitation has no bearing on the accuracy of the stress testing procedure implemented. The stress testing scope is limited to the CN. As such, an isolated/idealised radio environment, offered by the UERANSIM platform is ideal.

### B. *Core Network (CN)*

Our experiments leveraged the Open5Gs mobile core network implementation[10], largely because of its adoption coverage, wide community support, and support for the latest 3GPP specifications compared to its rivals. Open5Gs was designed following a cloud-native Service Based Architecture (SBA) [26] to enable concepts such as Network Slicing [27], and dynamics and efficient resource utilisation. For our testbed, we deployed the following network functions (all running on top of a Docker [28] container platform:

- **Access and Mobility Management Function (AMF)** — responsible for user mobility management, access authentication and authorisation;
- **User Plane Function (UPF)** — responsible for data packet inspection, routing, forwarding, and QoS handling;
- **Network Slice Selection Function (NSSF)** — responsible for Network Slice instances to serve a UE, by determining the allowed NSSAI (Network Slice Selection Assistance Information), and determining the AMF set to be used to serve the UE;
- and others, such as **Session Management Function (SMF)**, **Policy Control Function (PCF)**, **Authentication Server Function (AUSF)**, **Unified Data Management (UDM)** — for subscriber data and session management.

The core network, particularly the UPF network function constituted our system under test (SUT). The performance metric we measured is the percentage CPU utilisation by the UPF when subjected to the strain of user data forwarding emanating from multiple UE connections. In addition to

CPU utilisation due to user data, we also monitor the CPU utilisation resulting from control traffic during UE registration and attachment procedures (as per TS 23.502[29]).

### *C. Radio Access Network (RAN)*

The RAN domain is emulated using the UERANSIM stack running in 5G standalone (SA) mode. Like the UE domain, the emulated RAN protocol stack does not include the layers below the RRC layer. UERANSIM only supports a maximum number of 200 simultaneous UE connections. Beyond this limit, the base station starts to drop connection requests from the UE and throws a segmentation fault. As a result, we capped the number of connected UE processes to 200 and manipulated the traffic generator (which in our case was the standard Linux Ping Utility) to generate a large traffic volume. The variables manipulated on the traffic generated are the packet size and number of pings per UE process. Ideally, a traffic generator such as iPerf [30]/DTI-G[31] is preferred due to its superior (in terms of traffic volume) load-generating capabilities. Unfortunately, in our case, we wanted to instantiate traffic from inside of emulated UEs, through the GTP-U tunnels. This requires specifying individual UE process interfaces. To the best of our knowledge, this type of parametrisation, is only possible with the Ping traffic generator, which is why we opted to use this utility.

### *D. Computational Resources Allocations*

The computing resource allocation to the virtual machines hosting the 5G mobile network is broken down in Table II. The hardware specification of the commercial off-the-shelf (COTS) servers used to deploy OpenStack platform for hosting VMs is as follows: 1 TB RAM, 10 TB storage, and 208 virtual CPUs. The OpenStack environment is hosting a wide range of other virtual tenants, thereby limiting the amount of resources which could be allocated to our load testing testbed.

### *E. End-to-end Network Setup*

In order to establish an end-to-end 5G mobile network, certain configurations had to be made on each network domain. Table III below describes the key parameters configured to enable end-to-end 5G network connectivity. The connection between the UE and RAN domain was through a private network different from the subnet used to connect the RAN to the CN domain (see the top half of Figure 1). To minimise the propagation latency (for faster generation of results), our destination server for the ”pings” was an internal gateway directly connected to the CN via a public IP address.

## III. Stress Testing Procedure

Figure 2 summarises the procedure followed to stress-test the UPF implementation from the Open5Gs project. First, we deployed all the network components (i.e. UE, RAN, and CN) and made sure they were operational by checking the status of important services. Next, we sanitised each domain by killing all processes (such as HTTP processes) not needed to run the network components. This was followed by executing a batch UE provisioning script to add subscribers to the CN database. The CN network functions were then started, followed by the RAN network. To attach the RAN to the CN, the NSSAI of the RAN was inspected by the CN to make sure that it matches the AMF configuration. After a successful attachment of the RAN, multiple UE processes were instantiated and connection requests were forwarded over the RAN to the CN. The CN inspected the NSSAI passed by the UEs to ensure they matched. Upon successful NSSAI verification, the UEs were registered and multiple PDU sessions for user packet forwarding were established. This was followed by the parallel generation of traffic across the UEs, through the established PDU sessions to the Internet. The traffic generation was achieved by sending multiple “pings” to the UPF using the command “*Ping [parameters such as the number of packets]&*”. The packet size was kept at a default of 56 bytes. *Psrecord* [32] — a lightweight Python utility —

TABLE II: Compute resource allocation

| **Testbed Component** | **RAM** | **Virtual CPUs** | **Storage** | **Justification** |
|---|---|---|---|---|
| User Equipment | 4 GB | 4 vCPUs | 60 GB | The UEs do not run any other applications besides the Ping utility for traffic generation. The resource allocation was largely based on the fact that a single VM was used to host multiple instances of the UE running as a separate process, requiring more CPU allocation. |
| RAN/gNodeB | 8 GB | 4 vCPUs | 60 GB | The deployment guide does not specify minimum hardware requirements. Resource allocation decision was purely based on experience working with other radio network platforms such as OpenAirInterface5G and srsRAN. |
| Core Network | 8 GB | 4 vCPUs | 60 GB | This was based on the minimum required hardware specification for running Open5Gs CN. |

TABLE III: Mandatory configurations to establish an end-to-end 5G network connection

| **Parameter** | **Description** | **Configured Parameter** |
|---|---|---|
| PLMN | A combination of MCC and MNC values used for unique identification of operators | UE, RAN, and CN (AMF) |
| NSSAI | This is a combination of SST and SD values defined by 3GPP for distinguishing between different types of network slices (for example, SST 1 is used to identify a slice optimised for enhanced mobile broadband communication) | UE, RAN, CN (UPF, AMF, SMF) |
| TAC | An identifier of the location area within an operator’s network. | RAN, CN (AMF) |
| IMSI | A number that uniquely identifies every user in the network | UE, CN (UDM) |
| OPc, K | The parameters are used to authenticate users trying to access the network | UE, CN (UDM) |
| DNN | This is used to specify supported Data Networks (e.g. Internet, IMS, etc.) for each network slice. | UE, CN (UPF) |

was used to monitor and measure the CPU utilisation by the UPF network function as a result of both the User Plane (UP) and Control Plane (CP) traffic. After each traffic generation iteration, we made sure to kill all the UE, CN, and RAN processes before increasing the number of UEs for the next test iteration. The number of UEs was set to 50, 100, and then 200 UEs.

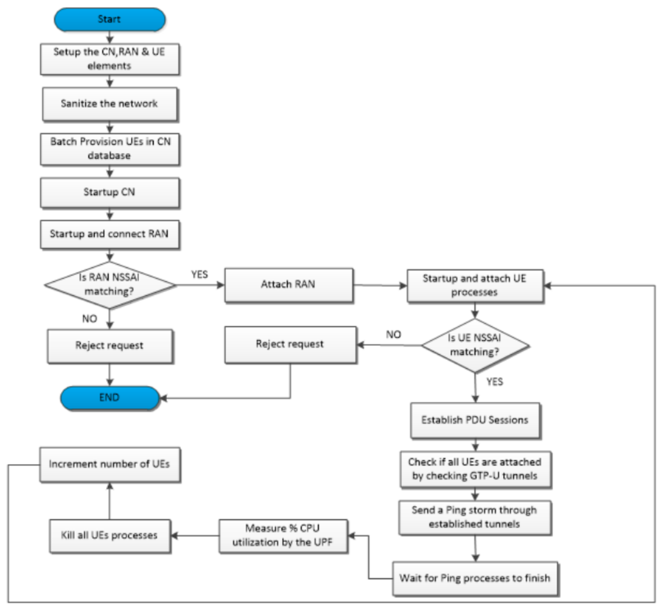
Start
Setup the CN,RAN & UE elements
Sanitize the network
Batch Provision UEs in CN database
Startup CN
Startup and connect RAN
Is RAN NSSAI matching?
YES
NO
Attach RAN
Startup and attach UE processes
Reject request
Reject request
NO
Is UE NSSAI matching?
YES
END
Establish PDU Sessions
Check if all UEs are attached by checking GTP-U tunnels
Increment number of UEs
Send a Ping storm through established tunnels
Kill all UEs processes
Measure % CPU utilization by the UPF
Wait for Ping processes to finish

Fig. 2: Summary of the testing procedure followed

## IV. Results and Discussions

The results from our stress testing efforts are as depicted in Figures 3–8. It may be noted that due to a scalability limitation of the UERANSIM platform, we were yet unable to push the theoretical load limit of the UPF, and could not quantify the absolute maximum traffic volume that would exceed the UPF CPU resources.

Next, it may be useful (for result interpretation) to highlight that the ping packets were effectively sent out concurrently, but only *approximately* simultaneously. In theory, the procedure sets the packets to be sent as follows (50/100/200 parallel series due to the respective numbers of UEs, with 100 sequential pings from each UE - the 200 UEs case shown; each ping packet denoted with a "*"):

*1)* * * * * * * * ...*(100 ping packets in series)*

*2)* * * * * * * * ...

*3)* * * * * * * * ...

...

*200)* * * * * * * ...

An expected ideal response to this load profile/"stimulus" should probably be in a series of high peaks in CPU usage at the time points of producing the ping packets.

In practice, there were slight variations of time between these pseudo-parallel executions of a ping and, more importantly, the ping utility itself waiting to send the next ping only after it receives a confirmation from the previous ping (or times out). This somewhat restricted the ability to saturate the control and data channels. Furthermore, as all the pings were the same and sent to the same interface, this must have forced the systems to use the same buffers and effectively serialise the processing of the ping packets. These effects would spread and widen the dome-like shapes created by the curves related to receiving the packets (e.g. CPU load).

It should also be further noted that since all processes (UE emulation, RAN and Core) were run on VMs and also on the same physical server, this may restrict the accuracy of timing the results.

Comparing the UPF CPU utilisation results in figures 6, 7, and 8 — for the user plane traffic — it is easy to observe that:

a) there is a trend to slightly increase the peak CPU usage as the number of UEs increases (from around 2% CPU usage for 1 ping to about 10% for 50 UEs to circa 15% for 100 UEs and to around 18% for 200 UEs);

b) the shape of the CPU usage curve for 50 UEs is fairly close to a rectangle, indicating that the system reasonably quickly responds to the traffic pattern and reaches the peak, then stays at the peak. The CPU usage goes down reasonably fast as well;

c) in contrast, the shape of the CPU usage curves for 100 UEs and especially for 200 UEs shows a slower incline and slower decline. This indicates that the system is adapting to the load slower and is likely buffering the packets for longer than should have been necessary;

d) furthermore, the overall width of the shapes created by the CPU usage curves increases, also supporting the indications that the system starts to take longer to process;

e) memory consumption increases only insignificantly and is probably an unimportant consideration for the given test scenario.

f) the total amount of time spent by the CPU processing user data packets for 50 UEs was about 20 s. In comparison, the CPU processing time for a single UE PDU session was found to be 2 s, which would theoretically translate to 25 s for 50 UEs, which is very close to the obtained CPU processing time for 50 UEs.

Comparing the UPF CPU utilisation results in figures 3, 4, and 5 — for the control plane traffic — it is easy to observe that:

a) for all control traffic-based CPU usage monitoring, a strong correlation can be observed between the number of UEs and the CPU packet processing time. For instance, it took about 10 s to process 50 UEs, 20 s for 100 UEs, and 60 s for 200 UEs. The spikes observed during the first couple of seconds are due to saturation on the control channel during the initial attach registration procedures.

b) the percentage CPU usage for 50, 100, and 200 UEs is virtually the same. This is likely due to retransmissions of the attach requests by the UEs after unsuccessful registration

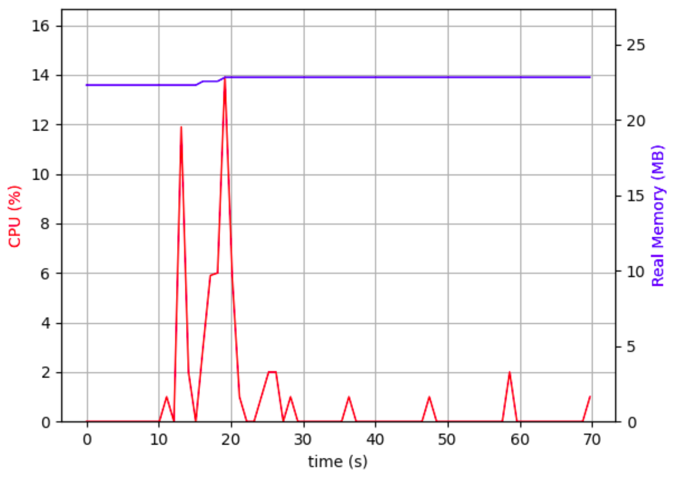

Fig. 3: UPF CPU Utilisation due to (CP) traffic from 50 UEs

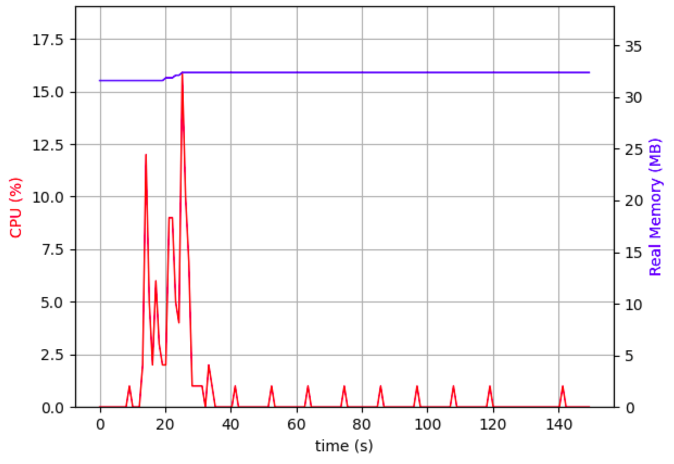

Fig. 4: UPF CPU Utilisation due to (CP) traffic from 100 UEs

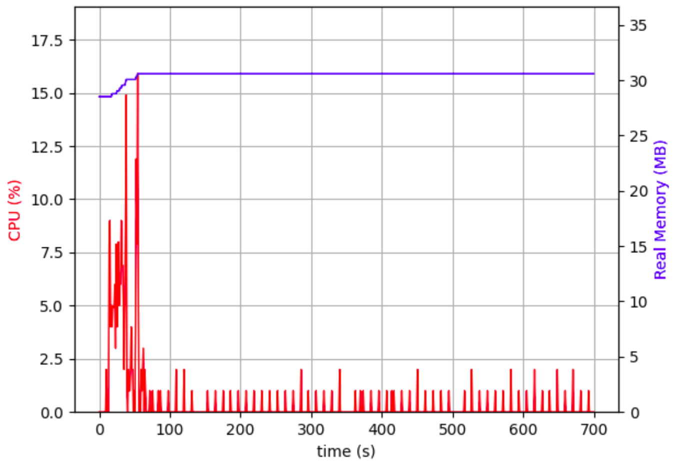

Fig. 5: UPF CPU Utilisation due to (CP) traffic from 200 UEs

attempts. At 50 UEs the number of the attach retransmissions were zero. However, for 100 and 200 UEs, the attach retransmissions increased significantly. There exists a strong correlation between the number of retransmissions, CPU usage, and CPU packets processing time distribution.

## V. Conclusion and Future Work

This paper proposed a test procedure for stress testing and profiling the performance of 5G Core Networks (CNs), with a particular focus on the User Plane Function (UPF), as the main node responsible for routing and forwarding user traffic to the Internet. The measured performance metric is the CPU usage efficiency of the UPF. The testbed deployed was based on open source 3GPP-compliant 5G User Equipment (UE), Radio Access Network (RAN), and CN stacks, namely, UERANSIM and Open5Gs. Our results show that the Open5Gs UPF implementation can achieve 200 parallel data connections using only 17% of the CPU and 200

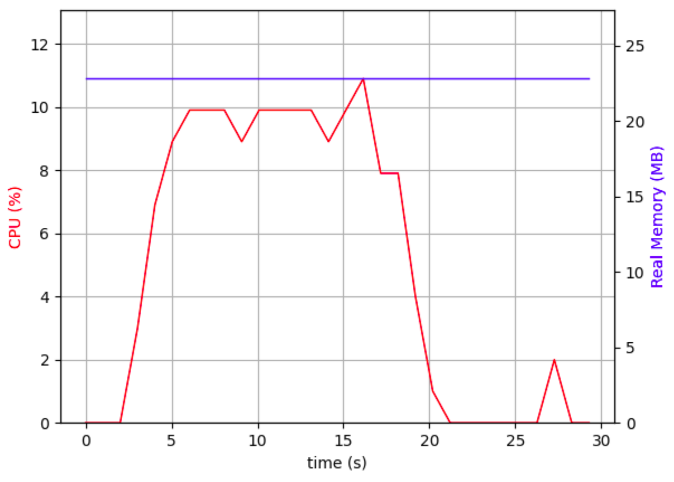

Fig. 6: UPF CPU Utilisation due to (UP) traffic from 50 UEs

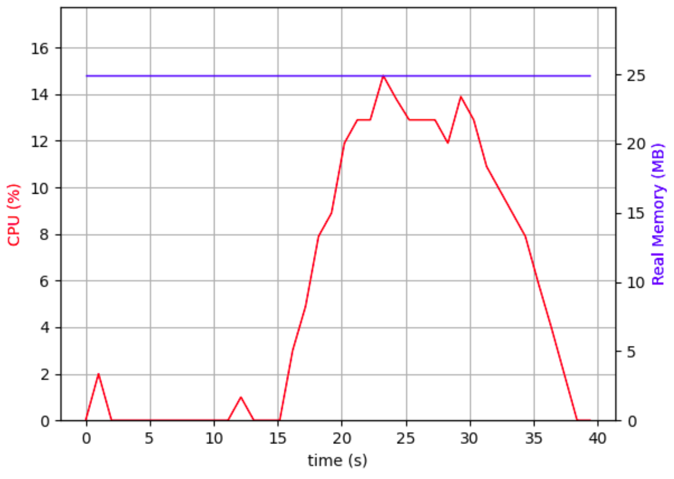

Fig. 7: UPF CPU Utilisation due to (UP) traffic from 100 UEs

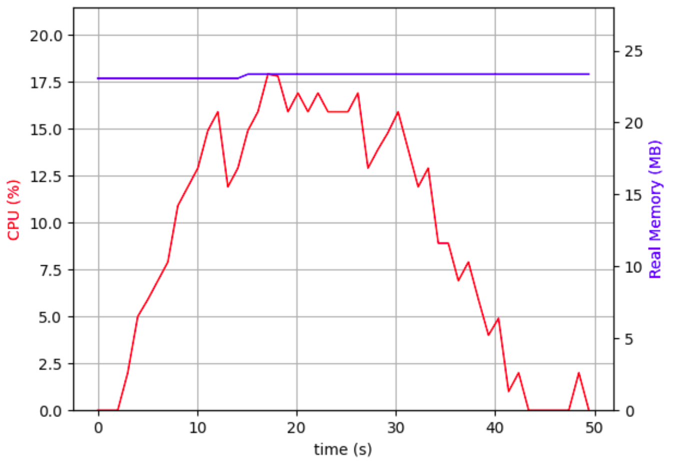

Fig. 8: UPF CPU Utilisation due to (UP) traffic from 200 UEs

control traffic connections using only 15% of the CPU.

As part of our future work, we intend to enhance the testbed scalability. This could include deploying multiple RANs or extending the UERANSIM implementation to support more concurrent UE connections, to achieve a more complete and comprehensive stress-testing solution. We also plan to optimise the resource utilisation efficiency (first for the Open5Gs UPF implementation, and then scale to other UPF solutions). This work is useful to network operators who have started implementing their 5G deployment strategies.

## References

[1] "ITU-R Report M.2370-0, IMT traffic estimates for the years 2020 to 2030, July 2015," accessed March 2023.

[2] "5G in Africa: realising the potential," 2022, Last accessed 20 February 2023. [Online]. Available: https://www.gsma.com/

[3] "5G standalone," 2022, Last accessed 09 March 2023. [Online]. Available: https://gsacom.com/

[4] H. Zhang, Z. Chen, and Y. Yuan, "High-performance upf design based on dpdk," in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*. IEEE, 2021, pp. 349–354.

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

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

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

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

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

[10] "Open5GCore," 2017, Last accessed 27 January 2023. [Online]. Available: https://www.open5gcore.org/

[11] "OpenAirInterface 5GC," 2019, Last accessed 03 March 2023. [Online]. Available: https://cumucore.com/mobile-private-network/

[12] "Use of open source software growing across telecom," 2018, Last accessed 13 March 2023. [Online]. Available: https://www.rcrwireless.com/20170120/featured/use-open-source-software-growing-across-telecom

[13] "Open source deployment," 2021, Last accessed 11 March 2023. [Online]. Available: https://www.ericsson.com/en/open-source

[14] "Introduction to open source private lte and 5G networks," 2022, Last accessed 08 March 2023. [Online]. Available: https://ubuntu.com/blog/introduction-to-open-source-private-lte-and-5g-networks

[15] "Cloud native design," 2022, Last accessed 13 March 2023. [Online]. Available: https://sdnfv.zte.com.cn/en/solutions/VNF/5G-core-network/cloud-native

[16] W.-E. Chen and C. H. Liu, "High-performance user plane function (upf) for the next generation core networks," *IET Networks*, vol. 9, no. 6, pp. 284–289, 2020.

[17] "The Pktgen Application," 2019, Last accessed 02 March 2023. [Online]. Available: https://pktgen-dpdk.readthedocs.io/en/latest/

[18] J. Zhou, Z. Ma, W. Tu, X. Qiu, J. Duan, Z. Li, Q. Li, X. Zhang, and W. Li, "Cable: A framework for accelerating 5G UPF based on eBPF," *Computer Networks*, vol. 222, p. 109535, 2023.

[19] M. A. Vieira, M. S. Castanho, R. D. Pacífico, E. R. Santos, E. P. C. Júnior, and L. F. Vieira, "Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–36, 2020.

[20] P. Enberg, A. Rao, and S. Tarkoma, "Partition-aware packet steering using xdp and ebpf for improving application-level parallelism," in *Proceedings of the 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms*, 2019, pp. 27–33.

[21] T. A. N. do Amaral, R. V. Rosa, D. F. C. Moura, and C. E. Rothenberg, "An in-kernel solution based on XDP for 5G UPF: Design, prototype and performance evaluation," in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 146–152.

[22] "MEC in 5G networks," 2018, Last accessed 13 March 2023. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[23] "Open Stack Neutron," 2019, Last accessed 07 March 2023. [Online]. Available: https://docs.openstack.org/neutron/latest/

[24] "UERANSIM," 2020, Last accessed 13 March 2023. [Online]. Available: https://github.com/aligungr/UERANSIM

[25] "5G-AN protocol stack," 2020, Last accessed 18 January 2023. [Online]. Available: https://www.lteprotocol.com/2020/02/5g-protocol-stack.html

[26] G. Brown, "Service-based architecture for 5G core networks," *Huawei White Paper*, vol. 1, 2017.

[27] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[28] "Docker engine," 2020, Last accessed 14 March 2023. [Online]. Available: https://docs.docker.com/get-docker/

[29] "Procedures for the 5G System," 2018, Last accessed 25 March 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/15.02.00_60/ts_123502v150200p.pdf

[30] "iPerf," 2016, Last accessed 12 March 2023. [Online]. Available: https://iperf.fr/

[31] "DTI-Gerf," 2014, Last accessed 12 March 2023. [Online]. Available: https://github.com/jbucar/ditg

[32] "psrecord," 2021, Last accessed 13 March 2023. [Online]. Available: https://pypi.org/project/psrecord/