# Investigating Inter-NF Dependencies in Cloud-Native 5G Core Networks

Endri Goshi*, Michael Jarschel†, Rastin Pries†, Mu He†, Wolfgang Kellerer*
*Technical University of Munich, †Nokia,
Munich, Germany
E-Mail:*{endri.goshi, wolfgang.kellerer}@tum.de, †{michael.jarschel, rastin.pries, mu.he}@nokia-bell-labs.com

***Abstract*—The increasing popularity of cloud-native approaches has led to their wide adoption in the telecommunications industry. 5G Core Networks (5GCN) are developed to take advantage of cloud-native design principles, with a high degree of functional decomposition and distributed deployment. This results in implications in inter-Network Function (NF) dependencies that need to be studied. This work focuses on investigating the effect that these dependencies have in how the resources are utilized from the 5GCN NFs.**

**We consider a private cloud environment where a reference 5G Core implementation, namely Free5GC, is deployed and orchestrated with Kubernetes. In addition, a gNB & UE Emulator is developed to allow for the execution of different control plane procedures. Our evaluations highlight the importance of catering for the inter-NF dependencies in achieving efficient resource utilization as well as avoiding deployments where a single NF can bottleneck the entire 5GCN.**

***Index Terms*—5G, Cloud-Native, VNF, Kubernetes**

## I. Introduction

Advancements in virtualization and softwarization technologies have enabled a paradigm shift in how mobile networks are designed and operated. High service availability, cost efficient operation and management, and on-demand service deployment are considered to be key aspects in developing future-proof 5th Generation (5G) and beyond mobile communication networks.

Following the trends in other information technology services, cloud-native deployments have gained increasing popularity in the field of telecommunications. Reasons for this being the high resiliency, high scalability and flexible orchestration that these environments offer. Therefore, mobile and wireless networks are leveraging cloud-native design approaches and are increasingly deployed in cloud environments.

In this regard, the 5G Core Network (5GCN) is defined as a Service-Based Architecture (SBA) [1]. Two key features of SBA are the high degree of functional decomposition and the distributed deployment of Network Functions (NFs). Therefore, adopting SBA results in tight dependencies between the NFs. Understanding these dependencies is a crucial task in order to achieve efficient resource allocation, avoid potential bottlenecks, improve scalability and optimize the scaling policies.

In light of this, in this work we investigate the inter-NF dependencies within 5GCN. For this, we orchestrate a Free5GC [2] deployment in a Kubernetes (K8s) [3] cluster and evaluate the resource utilization and Procedure Completion Times (PCTs) for *Registration* and *Deregistration* procedures under different CPU resource allocation scenarios.

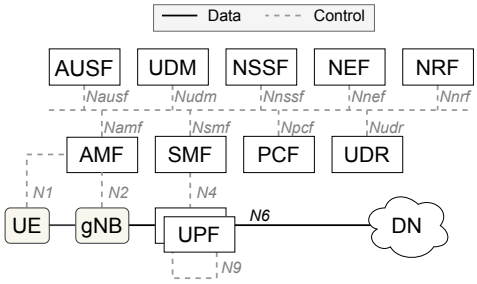


Fig. 1. 5G Core Service-Based Architecture. The dashed lines show control plane interfaces while continuous lines show data plane connections. In 5GCN control plane, inter-NF communication is performed using SBIs.

The remainder of this paper is structured as follows: Section II provides background information on 5G Core Networks and the tools utilized in this work. Related work is presented in Section III. Section IV describes the system set up for the evaluations, while Section V summarizes the evaluation results. Section VI concludes the paper.

## II. Background

### A. 5G Core Architecture

After several years of specification work, a standardized version of the 5G Mobile Core Architecture was introduced by 3GPP in Release 15 [1]. With scalability, flexibility, virtualization, and automation being the core principles, the 5G Mobile Core Network brings together two important concepts: i) Control and User Plane Separation (CUPS), and ii) Service-Based Interfaces (SBIs).

The increased number and types of connected user devices, and availability of high resource demanding applications have caused scalability and flexible deployment issues in the networks that feature integrated Control Plane (CP) and User Plane (UP) functions. CP-UP splitting has thus been introduced to address such issues firstly in the Evolved Packet Core (EPC) architecture in 3GPP Release 14 [4] where the Serving Gateway (SGW) and Packet Data Network Gateway (PGW) entities were decomposed into their respective control and user plane functions. Contrary to being an add-on feature in EPC, 5G is built on – and therefore mandates the adoption of – CUPS principles.

Simplification of inter-communication among Network Functions (NFs) has been another key aspect that has been tackled in 5GCN via the adoption of SBI. Use of SBI together with functional decomposition of the NFs has further evolved the architecture to a Service-Based Architecture (SBA) (see Fig. 1). In this concept, each NF exposes the set of services that it provides via standardized Application Programming Interfaces (APIs). These services can then be consumed by the other NFs through the advertised SBIs. The inter-NF communication therefore relies on the exposure and discovery of the services which is done with the help of a separate entity keeping registry for each service provided by NFs.

### B. Free5GC

Being one of the first, well-maintained, and widely used open-source implementations of the 5GCN SBA, Free5GC is considered in this work. Free5GC conforms to 3GPP Release 15 while being continuously developed to comply with 3GPP Release 16 [5] and beyond. Starting from v3.0.0, Free5GC offers a fully operational implementation of the Service-Based 5G Mobile Core. The NFs are developed as separate projects and the available CP NFs include AMF, AUSF, NRF, NSSF, PCF, SMF, UDM, and UDR, while a software implementation of the UPF is also provided. Alongside the CP NFs, a MongoDB [6] instance is deployed. This database is used to store NF-related information after they register with the NRF, as well as UE-related information such as authentication keys, authentication status, policy data etc. All the NFs in Free5GC can be compiled and deployed separately, making it a suitable candidate for evaluating the performance of distributed and cloud-native deployments of the 5GCN, as well as the inter-NF communication. To make the deployment of Free5GC easier in Kubernetes environments, we have contributed to its source code to enable the NFs to advertise Kubernetes Service IPs when registering to NRF.

### C. Kubernetes

As an open-source container orchestrator for cloud-native applications, Kubernetes (K8s) is the most used framework among public and private cloud operators for managing their clusters. Typically, a K8s cluster consists of one *Master* node and many *Worker* nodes. The Master has global view over the cluster it manages and stores the state of each of the Workers at all times. When new workload requests come from the operator, the Master takes care of scheduling them into the destination Workers. Flexible deployment of the workloads inside the cluster is one of the key advantages of using a container orchestrator such as K8s.

## III. Related Work

*1) Cloud-Native Mobile Core:* Following the design principles of cloud-native solutions, works [7] and [8] propose functionally decomposed architectures for the Mobility Management Entity (MME) in LTE networks. These microservice-oriented designs are superior to their monolithic counterparts with respect to resource efficiency, fault tolerance, scaling

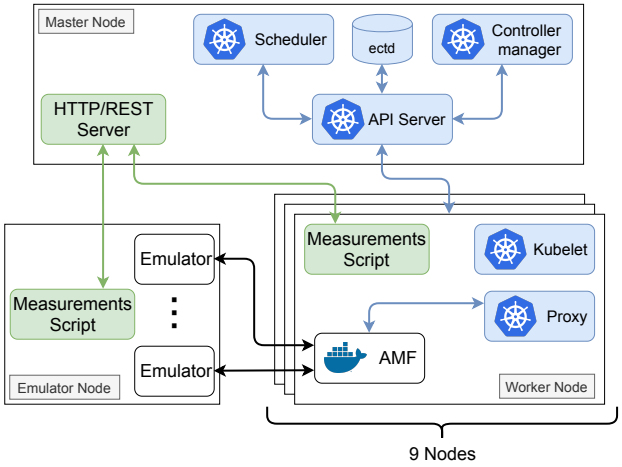


Fig. 2. System overview of our Free5GC deployment with Kubernetes.

and throughput. Authors in [9] propose a novel mobile core network architecture suitable for deployments in hyper-scale public clouds, promising higher availability than the considered telco solutions. Authors in [10] share their experience with respect to the cloudification of mobile networks and provide a comparison of the autoscaling mechanisms available in two widely used cloud orchestrates, namely Kubernetes and Mesos. In [11] the authors introduce Kube5G, a service platform for supporting cloud-native deployments of 4G/5G networks.

*2) SBA Evaluation:* Authors in [12] provide a prototype implementation of the 5G SBA using gRPC and propose a load balancing scheme for distributing the CP load across multiple NFs. In [13], a scaling mechanism based on Control Theory is proposed for the AMF in order to maintain optimal response times. However, the authors strictly focus on AMF and do not consider any other NFs or the impact of inter-NF communication. Lu et al. [14] propose an enhanced 5G SBA design aiming to improve the inter-NF communication by decoupling it from the business logic.

In comparison to state of the art, in this work we consider the 5GCN architecture as standardized in 3GPP Release 15 and focus on evaluating the inter-NF dependencies stemming from its distributed deployment in a private cloud environment orchestrated with Kubernetes. The control plane procedures are considered independently and the evaluations of resource utilization and latency are performed on a per-procedure basis.

## IV. System Design

### A. Framework overview

Fig. 2 offers an overview of the testbed deployed for the purpose of carrying out the evaluation of the inter-NF dependencies within 5GCN. The goal is to provide an environment similar to private cloud deployments, hence the selection of Kubernetes for orchestrating the Free5GC deployment. The testbed consists of a cluster of 11 hardware nodes, each used for one of the following purposes:

- *K8s Master and Measurements Sink Node* - As a K8s master, it manages the cluster of K8s workers and orches-

trates the deployment of NFs. Furthermore, it hosts an HTTP/REST server application that collects measurement data obtained from the Emulator and NF nodes.

- *5GCN NF Node* - Each control and user plane NF is deployed in a separate worker node. This approach i) avoids resource sharing between the NFs inside a single node, and ii) provides comparable network latency for NFs as the communication traverses the same network infrastructure. For the second point to hold true, the nodes are interconnected using a 1G switch, meaning that the deployment is done in an isolated environment. Since our 5GCN setup consists of 9 NFs (see Sec. II-B), a total of 9 NF Nodes are deployed.
- *gNB & UE Emulator Node* - The emulator tool instances are deployed in a separate node from the 5GCN NFs for the same reasons as explained above.

To ensure consistency throughout the evaluation process, the K8s Worker Nodes cluster (where the NFs are deployed) is homogeneous, consisting of DELL OptiPlex 9020 workstations equipped with an octa-core Intel i7-4770 CPU running at $3.40GHz$ and $16GB$ of RAM. All the nodes run Ubuntu 18.04.4 LTS with kernel version 5.0.0-23-generic. Furthermore, all the nodes run Docker v19.03.12 and Kubernetes v1.17.17. As for Free5GC, we deploy v3.0.5.

### B. gNB & UE Emulator.

The gNB and User Equipment (UE) emulator is a tool developed within the scope of this project for the purpose of evaluating the 5GC deployment. It is implemented in Go language and it allows the emulation of UE-related CP procedures for an arbitrary number of UEs. Currently, the tool supports the emulation of the following CP procedures: i) *Registration*, ii) *Deregistration*, iii) *PDU Session Establishment*, and iv) *PDU Session Release*.

The workflow of the emulation tool can be summarized in 5 steps, as follows:

1) The configuration file is parsed which defines the following types of parameters:
   - *Emulation-specific* parameters including the number of UEs to be emulated, the CP procedure type etc.
   - *5GCN-specific* parameters such as AMF and database IP addresses, Mobile Country Code (MCC), Mobile Network Code (MNC), etc.
   - *UE-specific* parameters such as ciphering and integrity algorithms.
2) The UE contexts are prepared and unique International Mobile Subscriber Identities (IMSIs) and authentication keys are generated, as well as unique identifiers for each UE within the gNB (`RAN_UE_NGAP_ID`) are assigned.
3) UE information is added to the database. This consists of information related to *Authentication*, *Access and Mobility*, *SMF Selection*, etc. This information is necessary as it is then queried by the 5GCN NFs during CP procedures.
4) Sequential execution of the CP procedures is performed. The execution is considered *sequential* because the emulator handles the procedures only for one UE at a time.

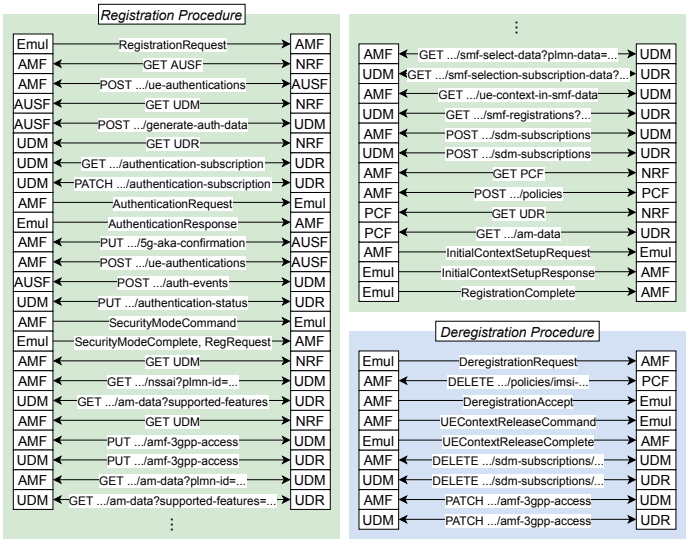


Fig. 3. Sequence diagram of the communication flows for the *Registration* and *Deregistration* procedures.

5) After the execution of the procedures for all the UEs has finished, Procedure Completion Times are reported to the HTTP/REST server for post-processing.

Regarding the sequential execution of the emulation, it was a design choice made due to an observed limitation of AMF in Free5GC. Initially the emulator executed CP procedures for many UEs simultaneously but with an increased degree of parallelism we observed a buffer overflow in the SCTP sockets as the AMF was unable to process all the incoming messages at once. This eventually led to the crash of AMF. To mitigate this problem without affecting the evaluation results, we scale the input traffic by simultaneously spawning multiple emulator instances with each of them executing CP procedures for one UE at a time.

### C. Control Plane Procedures

For the purpose of evaluation, in this work we consider two of the most common procedures in 5G, namely *Registration* and *Deregistration*. Both these procedures are initiated from the UE by sending a `Registration_Request` and `Deregistration_Request` respectively. After that, a series of inter-NF communication events is triggered within 5GCN, as illustrated in Fig. 3. Note that due to limited space, the interactions with the database are not depicted in this diagram. As mentioned in Section IV, the emulator is able to execute *PDU Session Establishment* and *PDU Session Release* as well. However, we observed that UPF was not able to handle the increased number of simultaneous procedures being emulated. Thus, they are left out of this work and will be investigated in future work.

## V. Evaluation, Results and Discussion

### A. Evaluation Methodology

For the performance evaluation we consider two deployment scenarios:

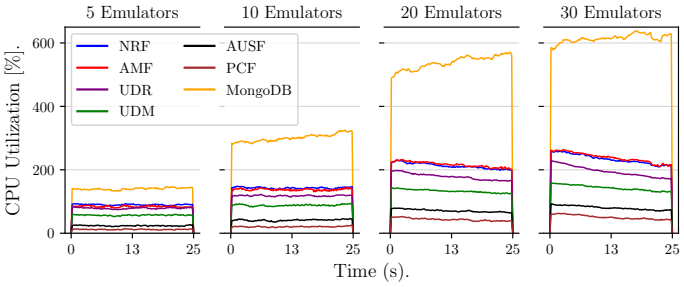


Fig. 4. CPU utilization of 5GCN NFs for *Registration* procedure for different number of gNB & UE Emulators running in parallel.

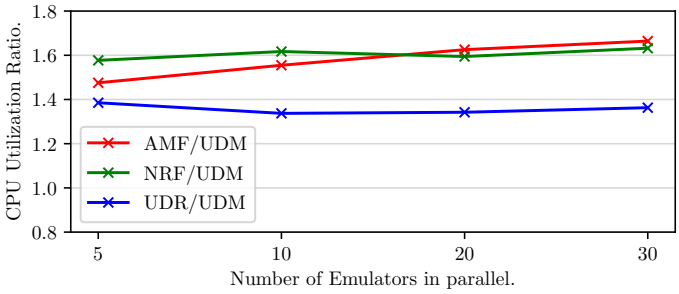


Fig. 5. Ratios of Average CPU Utilization values for *Registration* procedure w.r.t. the number of gNB & UE Emulators running in parallel.

- *Scenario I*: Baseline scenario where each NF has enough resources allocated to avoid becoming a bottleneck. No resource limitations are introduced during deployment and the NFs can utilize all the resources of the nodes that host them.
- *Scenario II*: A bottleneck is introduced by limiting the available resources of a single NF, part of the Service Function Chain (SFC) of the executed CP procedure.

Selected KPIs and their definitions are as follows:

- *CPU Utilization* - It is used for analyzing the CPU profiles of each 5GCN NF through the baseline scenario and consequently for deducing the values for the resource limitation in Scenario II. To measure the utilization of each of the NFs, a Python script that collects data every $200ms$ is deployed in each node. As the NFs are deployed in workstations with 8 CPU cores, the maximum possible CPU utilization is $800\%$. At the end of the execution, the measurements are reported to the *Sink Node*.
- *Procedure Completion Time (PCT)* - It is defined as the time it takes for the CP procedure to complete. The PCTs are measured from the emulator instances on a per-UE granularity. For the *Registration* procedure it represents the time from the moment `RegistrationRequest` is sent, until `RegistrationComplete` is acknowledged from AMF. For the *Deregistration* procedure it represents the time from the moment `DeregistrationRequest` is sent to AMF, until `UEContextReleaseComplete` is acknowledged from AMF (see Fig. 3).

Since most of the NFs are stateless, we observed that they showed a somewhat static memory usage; therefore, it is not considered as a KPI in this work. Network I/O, on the other hand, is an important KPI which can provide us useful information on how the NF's placement should be performed. Thus, it will be investigated in future work.

The measurements are performed for four different configurations, with the number of emulator instances varying between [5, 10, 20, 30]. To cater for the effect the database's size in the evaluation process, the number of UEs is kept constant at 4500 and they are distributed evenly among the spawned emulator instances; e.g., 5 emulators with 900 UEs/emulator, 10 emulators with 450 UEs/emulator, etc. The number of these emulator instances corresponds to the number of UEs being served simultaneously.

### B. CPU Utilization Profiles

As illustrated in the sequence diagrams in Fig. 3, different CP procedures trigger different NFs. This can be seen also from the CPU utilization profiles shown in Fig. 4 and Fig. 6 for the *Registration* and *Deregistration* procedures respectively. As MongoDB utilization quickly reaches $\sim 600\%$ (see Fig. 4), we do not further scale the input traffic in order to avoid scenarios where MongoDB would become a bottleneck. As the CP procedures have different computational demands, these profiles differ not only in terms of the involved SFCs but also on the resource utilization of individual NFs.

To serve the *Registration* procedure, a big SFC is triggered inside 5GCN, with NRF and AMF being the most critical NFs. Fig. 4 shows that the increased number of simultaneously-served UEs results in increased CPU utilization of the NFs. Furthermore, due to the tight inter-NF dependencies, the CPU Utilization profiles seem to follow the same trend among the different configurations. To better illustrate this, we refer to Fig. 5 where the ratios of the average CPU utilization of the NFs for the *Registration* procedure are shown. AUSF and PCF are not considered in this plot because their utilization is low and small fluctuations can lead to inaccurate observations. Therefore, we consider UDM as the baseline to calculate the ratios. We observe from the plot that there are only small fluctuations of the ratios as we scale the number of UEs being served in parallel, thus confirming that the 5GCN NFs are tightly coupled. For this reason, we argue that when orchestrating cloud-native deployments of 5GCN, the SFC should be considered together, rather than orchestrating separate NFs.

Similarly, we refer to Fig. 6 which illustrates the CPU utilization profiles for the *Deregistration* procedure. Comparing the measurements with Fig. 4, we see that the *Deregistration* procedure triggers a smaller and less-demanding SFC resource-wise. Nonetheless, while the input traffic is scaled up by deploying more emulators in parallel, the inter-NF utilization ratios are maintained, as shown in Fig. 7. If we compare these ratios with the ones in Fig. 5, we see that there are differences between ratios of the same NFs, e.g. *AMF/UDM* is $\sim$1.6 for *Registration* and $\sim$2.2 for *Deregistration*. Therefore, we argue that the orchestration of cloud-native deployments would benefit from a per-procedure SFC orchestration of the 5GCN. In such scenarios, 5GCN can be deployed as a set of SFCs, each of which would handle only one type of CP

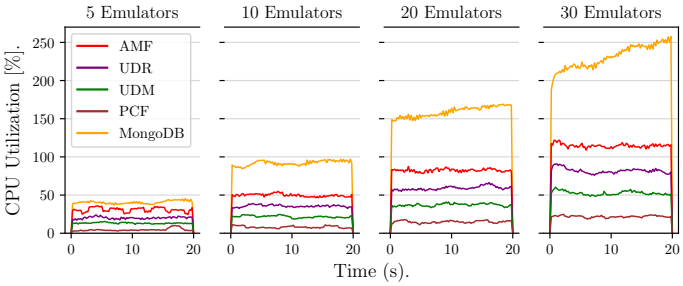


Fig. 6. CPU utilization of 5GCN NFs for *Deregistration* procedure for different number of gNB & UE Emulators running in parallel.

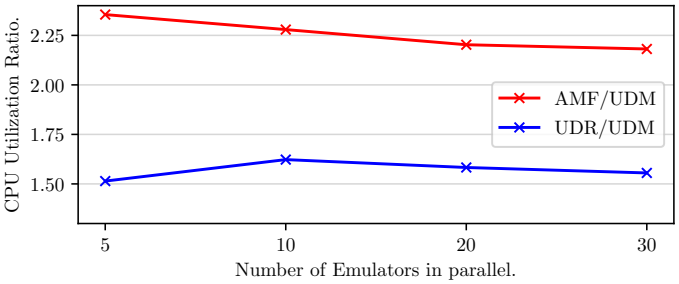


Fig. 7. Ratios of Average CPU Utilization values for *Deregistration* procedure w.r.t the number of gNB & UE Emulators running in parallel.

procedure. The resource allocation can then be done based on the requirements of the procedure they serve. This way we can obtain more granularity when allocating the resources and avoid single-NF bottlenecks within the SFCs.

### C. Impact of a Bottlenecked NF

Since the NFs demonstrated a highly correlated CPU utilization, we evaluated the impact that a single bottlenecked NF can have on the performance. Therefore, we artificially introduce a bottleneck by limiting the available CPU resources of one NF. Fig. 8 shows how the PCTs for *Registration* procedure change for two different deployments, where: i) no bottleneck in introduced in the network, and ii) UDM's CPU resources are limited to 1 core. This limit corresponds to the average utilization value observed during the execution of *Registration* procedure with 10 emulators deployed in parallel. We observe that by limiting the resources of a single NF we can bottleneck the entire 5GCN and considerably degrade its performance, while also leading to inefficient resource utilization as the other NFs cannot take advantage of their available resources. The above scenario can be avoided if the inter-NF resource utilization ratios are taken into consideration during the resource allocation process. Measurements were performed also for configurations where other NFs were limited but due to space limitations they are left out of this work. However, the same behavior as in Fig. 8 was observed.

## VI. Conclusion and Future Work

The focus of this work is the investigation of inter-NF dependencies in 5GCN resulting from the adoption of SBA. For this we consider a private cloud deployment of Free5GC, orchestrated with Kubernetes. For the purpose of

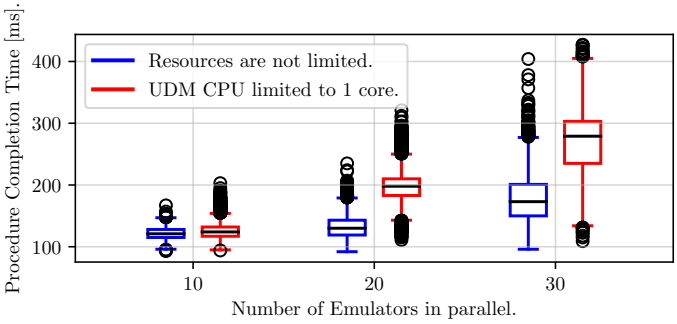


Fig. 8. Comparison of PCTs for *Registration* procedure for scenarios where: i) no limitation of resources is performed during deployment, and ii) UDM CPU resources are limited to $1000m$ (1 Core). This limitation causes a bottleneck in the 5GCN leading to decreased performance and increased PCTs.

evaluating the deployment, a gNB & UE Emulator able to execute different control plane procedures was developed. In the evaluated control plane procedures, we observed a very high correlation in resource utilization between the NFs that took part. Furthermore, we showed that incorrect resource allocation can lead to any NF becoming a bottleneck, and consequently degrading the performance of 5GCN as a whole.

As a future work, we aim to tackle these problems by developing mechanisms that deploy and orchestrate 5GCN on a per-procedure basis. Moreover, we will evaluate SFC scaling mechanisms that try to address inter-NF dependencies and compare them with baseline NF scaling implementations.

## References

[1] 3GPP, "3GPP TS 23.501 - System architecture for the 5G System (Rel 15)," 2021.
[2] "Free5GC," https://www.free5gc.org/, 2021.
[3] "Kubernetes," https://kubernetes.io/, 2021.
[4] 3GPP, "3GPP TS 23.002 - Network architecture (Rel 14)," 2017.
[5] ——, "3GPP TS 23.501 - System architecture for the 5G System (Rel 16)," 2021.
[6] "MongoDB," https://www.mongodb.com/, 2021.
[7] V. Nagendra, A. Bhattacharya, A. Gandhi, and S. R. Das, "Mmlite: A scalable and resource efficient control plane for next generation cellular packet core," in *Proceedings of the 2019 ACM Symposium on SDN Research*, ser. SOSR '19, New York, NY, USA, 2019, p. 69–83.
[8] P. C. Amogh, G. Veeramachaneni, A. K. Rangisetti, B. R. Tamma, and A. A. Franklin, "A cloud native solution for dynamic auto scaling of mme in lte," in *2017 IEEE 28th Annu. Int. Symp. on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1–7.
[9] B. Nguyen, T. Zhang, B. Radunovic, R. Stutsman, T. Karagiannis, J. Kocur, and J. Van der Merwe, "Echo: A reliable distributed cellular core network for hyper-scale public clouds," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2018, p. 163–178.
[10] D.-H. Luong, H.-T. Thieu, A. Outtagarts, and Y. Ghamri-Doudane, "Cloudification and autoscaling orchestration for container-based mobile networks toward 5g: Experimentation, challenges and perspectives," in *2018 IEEE 87th Vehicular Tech. Conf. (VTC Spring)*, 2018, pp. 1–7.
[11] O. Arouk and N. Nikaein, "Kube5g: A cloud-native 5g service platform," in *GLOBECOM 2020*, 2020, pp. 1–6.
[12] T. V. Kiran Buyakar, H. Agarwal, B. R. Tamma, and A. A. Franklin, "Prototyping and load balancing the service based architecture of 5g core using nfv," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 228–232.
[13] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin, and D. Darche, "On the scalability of 5g core network: The amf case," in *2018 15th IEEE Annu, Consumer Commu. Netw. Conf. (CCNC)*, 2018, pp. 1–6.
[14] J. Lu, L. Xiao, Z. Tian, M. Zhao, and W. Wang, "5g enhanced service-based core design," in *2019 28th Wireless and Optical Communications Conference (WOCC)*, 2019, pp. 1–5.