Open5GLoS: Scalable Cloud-Native Architecture for Open-Source 5G Core Networks

Quang-Huy Tran*†, Quang Tung Thai†, and Namseok Ko*†
* Departments of Information and Communication Engineering, University of Science and Technology (UST), Daejeon 34113, South Korea
†*Mobile Core Network Research Section, Network Research Division*
Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea
{tqhuy, tqtung, nsko}@etri.re.kr

***Abstract*—This paper presents Open5GLoS, the first proof-of-concept (PoC) framework enabling horizontal scaling and NGAP-aware load balancing of the Access and Mobility Management Function (AMF) in a Kubernetes-based open-source 5G Core. Built on Free5GC, Open5GLoS addresses three limitations in current 3GPP architectures: static AMF ID assignment, gNB–AMF connectivity issues from dynamic IPs, and the absence of standardized SCTP/NGAP load balancing. Our design generates unique AMF IDs, provides gateway-based gNB–AMF discovery, and applies UE-aware load balancing at the NGAP layer. Experiments show an average 15–30% reduction in UE registration time for up to 200 UEs, peaking at 36.9% under higher loads, with 100% registration success even at 250 UEs where single-AMF deployments fail. These results highlight architectural gaps in the 3GPP-defined 5G Core, particularly in identity management, service discovery, and transport protocols at scale. By delivering measurable performance gains, Open5GLoS serves as both a practical reference for cloud-native 5G deployments and a critical input for future standardization.**

***Index Terms*—5G Service-Based Architecture (SBA), 5G Core Network, AMF, Scaling, Load Balancing.**

I. Introduction

The evolution of mobile networks toward 5G has brought fundamental changes to the design, deployment, and operation of core network functions [1], [2]. A key enabler for meeting the scalability, flexibility, and operational efficiency demands of next-generation networks is the adoption of cloud-native deployment models. By leveraging containerization and orchestration platforms such as Kubernetes, 5G Core networks can dynamically adapt to traffic fluctuations, accelerate service innovation, and reduce operational costs [3], [4].

Although 3GPP has introduced the service-based architecture (SBA) to modernize the 5G Core, a clear gap remains between traditional telecom standards and cloud-native design principles. Legacy implementations often rely on monolithic, stateful components and rigid interfaces, which hinder scalability and elasticity. In contrast, cloud-native systems emphasize stateless microservices, API-driven communication, and declarative state management, enabling more resilient and agile network functions (NFs). Bridging this gap is essential to fully realize the potential of 5G and beyond.

Within the 5G Core, the Access and Mobility Management Function (AMF) is a critical control-plane NF responsible for UE registration, connection management, and mobility control. Despite its importance, current AMF implementations face significant limitations in horizontal scaling and load balancing. When UE demand surges, insufficient scaling support can degrade performance and user experience. Moreover, secure and efficient interconnection between the radio access network (RAN) and the core must be maintained even under dynamic scaling conditions.

To address these challenges, we propose Open5GLoS, the first proof-of-concept (PoC) framework for horizontally scalable, NGAP-aware load-balanced AMF deployment in a Kubernetes-based open-source 5G Core. Our contributions are summarized as follows:

- First scalable AMF PoC on Free5GC: Implements dynamic horizontal scaling of AMFs within a Kubernetes environment, ensuring unique AMF IDs for each instance.
- NGAP-aware load balancing: Introduces a UE-level load balancing mechanism that considers NGAP context to distribute traffic evenly across AMFs.
- Resolution of cloud-native integration issues: Addresses dynamic IP handling and gNB–AMF discovery challenges through a gateway-based approach.
- Performance validation: Demonstrates up to 36.9% reduction in UE registration time under high load and 100% registration success in scenarios where single-AMF deployments fail.

By closing the gap between 3GPP-defined architecture and cloud-native design, this work offers a practical solution for improving 5G Core scalability and resilience, while providing insights for future 5G/6G standardization efforts.

The remainder of this paper is organized as follows. Section II presents a background and overview of existing works. Section III discusses the current stage of open-source architecture with its challenges and presents the proposed PoC. The experimental results are mentioned in Section IV. Finally, Section V concludes the paper with future research directions.

II. Background

A. 5G Core Architecture and AMF Specifications

The 5G core network (5G CN), as defined by 3GPP, adopts the SBA consisting of discrete NFs that communicate via standardized interfaces [2]. A central element in this architecture

is the AMF, where key 3GPP specifications relevant to AMF include:

- AMF ID: Each AMF instance is uniquely identified within the network, allowing for efficient routing and management of signaling messages.
- Interfaces: The AMF communicates with various network functions via interfaces such as N1 (UE), N2 (RAN), N11 (SMF), and others, supporting both control and management tasks [5].
- NGAP over SCTP usage [6]: Next Generation Application Protocol uses Stream Control Transmission Protocol (SCTP) to transmit messages over the transport layer for signaling between gNB and AMF, providing reliable message delivery, multi-homing, and multi-streaming capabilities essential for 5G control plane operations.

While 3GPP specifications define the functional architecture of 5GC, they do not explicitly prescribe the underlying deployment model. This leaves implementation-specific details to vendors or operators.

B. Open Source 5G Core Implementations and Limitations

Two well-known open-source 5G core implementations, Free5GC and Open5GS, are designed to facilitate research, experimentation, and small-scale deployments [7] and [8]. For example, Free5GC supports integration with the RAN emulator and offers a modular implementation of 3GPP-specified core functions. It supports containerized and Kubernetes-based deployments, aligning with cloud-native principles to some extent. Despite these advancements, there are several limitations. Statelessness helps cloud-native systems scale, but many 5GC functions, like AMF, still have stateful components, which makes failover and scaling more difficult. Furthermore, most open-source 5GC projects do not fully implement dynamic service discovery, which is essential for microservices and restricts true cloud-native operation. While Kubernetes makes pod scaling possible, the advantages of 5G CN are hampered by the fact that control plane functions such as AMF in Free5GC are not naturally suited for horizontal scaling or smooth load balancing.

C. Prior Research about Scaling and Load Balancing

Several research efforts have addressed scaling and load balancing for the 5G core network. Three parameters were used by the authors of [9] to propose dynamic control plane load balancing techniques: load on the AMF, service time, and the pending number of requests at the AMF. Similarly, [10] created a probability module to reroute the subsequent UEs that arrived. Using RAN slice selection, Buyakar et al. [11] created a prototype for auto-scaling. Based on the CPU usage of the related network functions, Bello et al. [12] suggested a predictive autoscaling method for the evolved packet core (EPC). The author of [13] also suggested load balancing and auto-scaling for the user plane gateways in the 5G network. Based on the network load, they integrated load balancing and auto-scaling to provide effective user plane services. Nevertheless, those works are not implemented in any open-source 5GC system; instead, they are evaluated in virtual network functions (VNFs) or on local computers. In [14], Kubernetes is used to deploy 5G CN with load balancing and auto-scaling through the use of LoxiLB, an eBPF-based L4/NAT load balancer [15]. LoxiLB listens on a virtual SCTP IP (VIP) and forwards connections to AMF pods; in "one-arm" NAT mode it replaces the VIP with the chosen AMF's IP. However, when a session is created between a gNB and an AMF, all UEs belong to the gNB are served by that AMF. This results in an imbalance in load between AMFs since the number of UEs at each gNB is unknown.

Despite extensive research on load balancing and scaling for AMF, no open-source frameworks that are widely used yet show true horizontal scaling of AMF in a cloud-native (Kubernetes) environment. Furthermore, the majority of research and implementations on load balancing and scaling concentrate on local host deployments or VNFs, failing to fully utilize the automation and elasticity of cloud-native platforms. Finally, the UE load is not taken into account by these load balancers at the NGAP level message. Therefore, in order to construct the first PoC at the 5G control plane, we reexamine the load balancing and scaling problem.

D. Summary and Research Gap

Existing open-source 5GC platforms and prior research show that while scaling and load balancing mechanisms exist, they do not fully address the challenges of horizontally scaling AMF in a Kubernetes-based 5GC with NGAP-aware UE-level load balancing. In particular, no widely adopted open-source framework currently provides:

- Dynamic, unique AMF ID assignment during scaling.
- Seamless gNB–AMF discovery in the presence of dynamic IPs.
- UE-aware NGAP load balancing for equitable traffic distribution.

This gap motivates the development of Open5GLoS, which implements all three capabilities within a reproducible, open-source PoC.

III. System Design and Open5GLoS Framework

This section presents the overall architecture of the proposed system, the challenges in achieving scalable AMF deployments in a cloud-native 5GCN, and the design of the Open5GLoS framework that addresses these challenges.

A. System Architecture

The core backbone of the 5G Core Network (5GCN) is deployed in a cloud-native environment, primarily using Kubernetes. Kubernetes provides dynamic scheduling, service discovery, and lifecycle management for containerized applications. In our setup, each 5G Core Network Function (NF) runs as an independent container within a Kubernetes Pod. Pods are isolated logical units, and Kubernetes Services expose groups of Pods via stable virtual IPs, with DNS-based service discovery. Container Network Interface (CNI) plugins handle

low-level pod-to-pod and pod-to-service connectivity (e.g., IP address assignment).

In the open-source Free5GC implementation, the following adjustments were made for Kubernetes deployment:

Configuration file management: NF-specific configuration files define network parameters, interface addresses, and service endpoints. These are managed through Kubernetes ConfigMaps and Secrets, allowing dynamic updates without Pod restarts.

Service registration and discovery: Each NF registers with the Network Repository Function (NRF) using Kubernetes-resolvable service names (e.g., http://amf.namespace.svc.cluster.local), enabling inter-NF communication without static IPs.

Deployment state: NFs that store persistent data, such as the Unified Data Repository (UDR) and Unified Data Management (UDM), use persistent volumes or external databases (e.g., MongoDB in Free5GC). Kubernetes can create multiple replicas of an NF such as the AMF using the same configuration but with distinct Pod names.

B. Design Challenges

Deploying scalable and dynamic AMFs with load balancing in Kubernetes introduces several technical challenges, especially when reconciling 3GPP architectural assumptions with cloud-native behavior.

1) AMF ID Uniqueness: 3GPP specifications require each AMF instance to be uniquely identified via an AMF ID, which is used in registration and communication with gNBs and UEs. Since the AMF assigns the Globally Unique Temporary UE Identity (GUTI), a unique AMF ID is necessary to maintain UE context. Existing open-source deployments often assume a single predefined AMF identity, causing conflicts when scaling, leading to re-authentication failures. We address this by generating unique AMF IDs in an init container at Pod startup, embedding part of the Pod hostname into the AMF configuration.

2) Dynamic IPs and gNB Discovery: Kubernetes assigns ephemeral IP addresses to Pods, but RAN simulators such as UERANSIM require static AMF IP configuration [16]. While AMF replicas are dynamic, gNBs expect fixed endpoints, risking disconnections. Our solution uses a gateway-based load balancer targeting all AMFs within a Kubernetes Service, maintaining NGAP session stickiness for continuity while distributing new UEs across AMFs.

3) SCTP Load Balancing: Kubernetes lacks native SCTP load balancing, necessitating an L4/L7 load balancer with SCTP support. At L4, SCTP association requests from gNBs are distributed evenly across AMFs, but all UEs from a gNB remain bound to its initial AMF, potentially causing load imbalance. In contrast, L7 load balancing at the NGAP level makes per-UE decisions, selecting AMFs based on UE session context and current load, thus achieving finer-grained distribution and improved scalability.

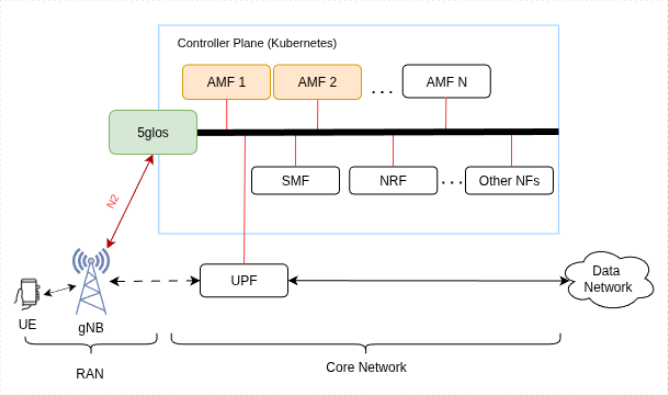


Fig. 1: System architecture view of the 5G network with Open5GLoS in the cloud

C. Open5GLoS Framework Design

To address the identified challenges, the Open5GLoS proof-of-concept integrates cloud-native deployment capabilities with NGAP-aware load balancing. The framework consists of the following core components:

- **Horizontally scalable AMF deployment:** Each AMF instance is assigned a unique AMF ID at startup, enabling dynamic scaling without identity conflicts. Scaling decisions can be triggered manually via the Kubernetes command-line interface or automatically based on UE load thresholds.
- **NGAP-aware load balancing:** The gateway load balancer assigns UEs to AMFs according to a configurable load-availability policy at the NGAP level, distributing control-plane load more evenly across AMF instances.
- **UE Context Management:** The gateway maintains a UE context list containing mappings of LB NGAP UE ID, destination AMF ID, AMF NGAP UE ID, RAN ID, and RAN NGAP UE ID. This mapping enables seamless interception and routing of NGAP messages between UEs, gNBs, and AMFs without disrupting session continuity. The list is dynamically updated as UEs connect, disconnect, or move between gNBs, ensuring that load-balancing decisions always reflect the current network state.
- **Dynamic Scaling Integration:** AMF resource utilization is monitored through Kubernetes metrics. When CPU or memory usage exceeds configurable thresholds, new AMF Pods are automatically deployed. Under low-load conditions, surplus AMF instances are terminated to conserve resources. This elasticity allows the framework to handle traffic bursts efficiently while minimizing idle capacity.

In our implementation, AMF configuration files are generated from a template and customized at Pod initialization by a script running in an init container. This script derives a unique AMF ID from the Pod name, ensuring identity uniqueness even during scale-out or scale-in operations. For

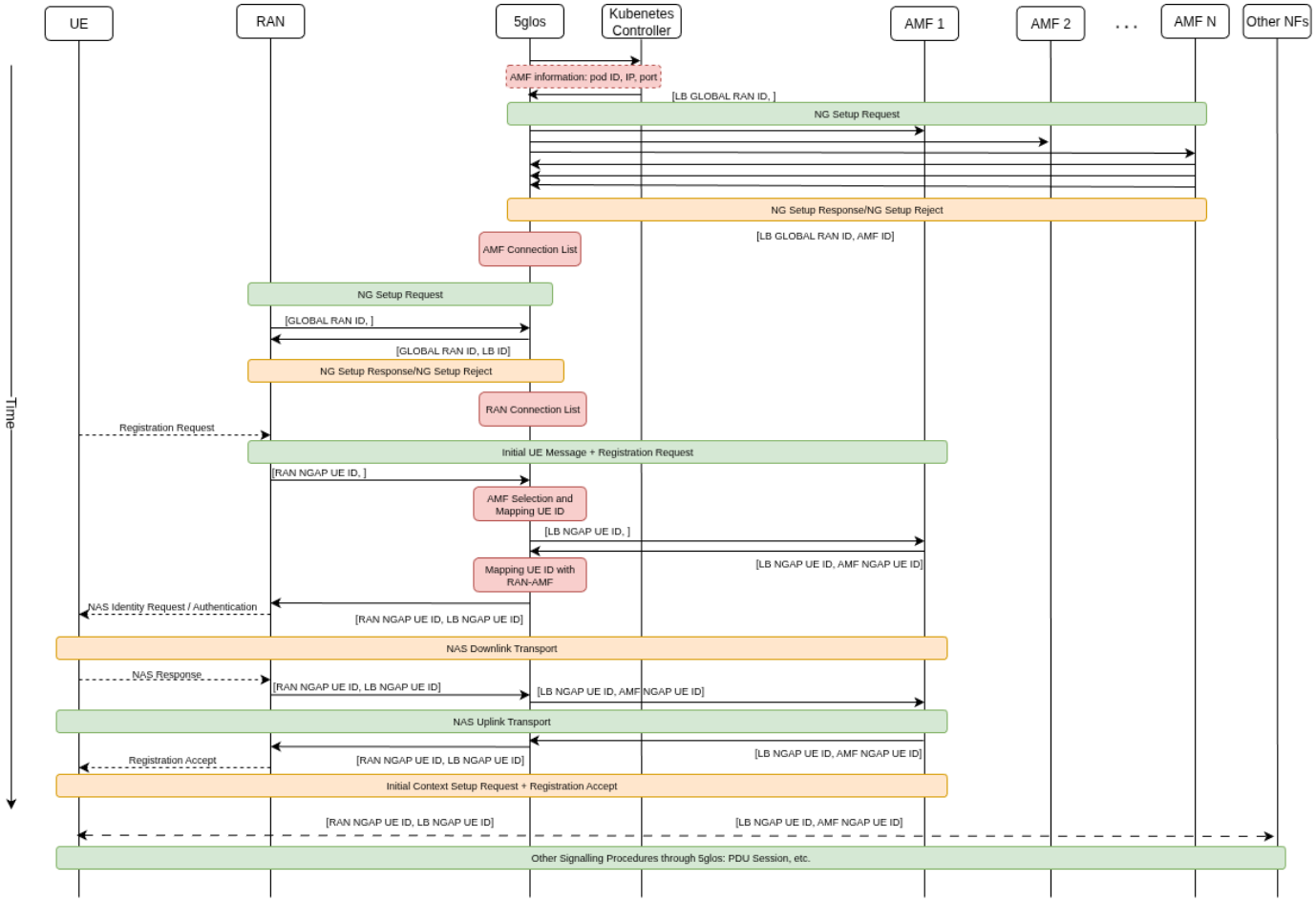


Fig. 2: Open5GLoS workflow from gNB registration to UE message handling

dynamic IP management and gNB discovery, the gateway exposes a stable service endpoint to gNBs and internally routes traffic to the selected AMF Pods based on the NGAP-aware algorithm. Figure 1 shows the placement of the gateway and AMF instances in the Kubernetes-based 5GCN.

D. Operational Flow

Figure 2 illustrates the full message flow:

- **Step 1 – NG Interface Setup:** gNBs connect to the gateway and send an NGSetup Request to establish the NG interface. Upon verification, the gateway adds the gNB to its list and replies with an NGSetup Response.
- **Step 2 – AMF Discovery:** The gateway queries Kubernetes for active AMF Pods and sends NGSetup Requests to each. Successful responses are stored with AMF IDs and Pod IDs.
- **Step 3 – UE Registration and Load Balancing:** When a UE registers, the gNB sends an NGAP Initial UE Message to the gateway. The gateway inspects the message, applies the load-balancing algorithm, selects an AMF, and rewrites identifiers before forwarding.
- **Step 4 – Downlink Handling:** For downlink NGAP messages, identifier mappings are reversed to deliver them to the correct gNB.

E. Key Benefits of the Open5GLoS Framework

The proposed Open5GLoS framework offers the following key advantages over existing open-source 5GCN deployments and prior research:

- **True horizontal scalability for AMF:** Supports on-demand scale-out and scale-in of AMF instances with unique AMF IDs, ensuring state consistency and avoiding re-authentication failures.
- **NGAP-aware UE-level load balancing:** Achieves finer-grained traffic distribution by making load-balancing decisions at the UE session level, preventing imbalance caused by gNB-level association in conventional L4 methods.
- **Elastic resource utilization:** Integrates Kubernetes-based metrics monitoring to dynamically provision or release AMF instances according to traffic load, improving efficiency under variable demand.
- **Cloud-native integration:** Operates seamlessly within Kubernetes environments, enabling reproducible deploy-

ment, automated service discovery, and compatibility with CI/CD workflows for 5GCN research and operations.

- **Improved performance and reliability:** Experimental results show reduced UE registration time (up to 36.9% improvement under high load) and 100% registration success in scenarios where single-AMF deployments fail.

IV. Network Setup, Implementation, and Result Analysis

A. Experimental Environment and Deployment

To evaluate the Open5GLoS framework, we conducted experiments in both inside- and outside-Kubernetes scenarios. The 5G Core Network (5GCN) was deployed using the open-source Free5GC [7] within a Kubernetes cluster, while the Radio Access Network (RAN) and UEs were emulated using UERANSIM [16] running outside the cluster.

Hardware platform: All experiments were executed on a physical server equipped with an Intel i7-10700K CPU, 32 GB of RAM, and running Ubuntu 24.04 LTS. The Kubernetes node was configured using a Docker driver with 4 vCPU cores, 8 GB of memory, and 50 GB of disk storage.

Software stack:

- Free5GC v4.0.1 (control-plane functions: AMF, SMF, UPF, NRF, etc.)
- Kubernetes (minikube) v1.36.0 with CNI plugin (Calico/Flannel) for pod networking.
- UERANSIM v3.2.7 for gNB and UE simulation.
- Open5GLoS implementation integrated as a gateway between the RAN and the Kubernetes-based 5GCN.

Deployment topology: AMF, SMF, UPF, NRF, UDR, and UDM were deployed as independent pods. AMF replicas were scaled dynamically according to the test scenario (1 AMF vs. 5 AMFs). The gateway provided a single stable endpoint to gNBs and internally routed traffic to AMFs using NGAP-aware load balancing.

The complete Open5GLoS implementation, including configuration scripts, load-balancing modules, and deployment guidelines, is publicly available [17] to enable reproducibility and extension by the research community.

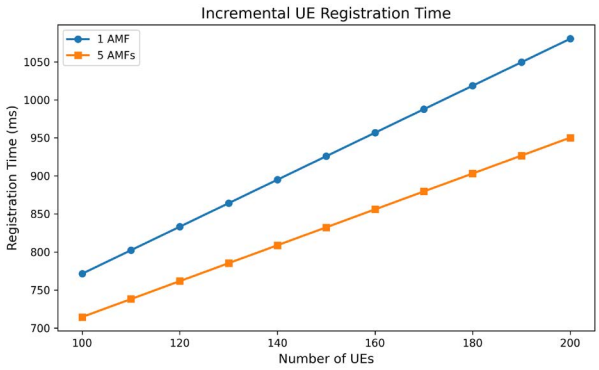
B. Test Scenarios

Two scenarios were designed to evaluate scalability and performance:

Scenario 1 – Fixed UE Load: UE count was varied between 100 and 250 in steps, with all UEs connecting simultaneously to the gNB. This scenario measures registration time under static load conditions for both 1 AMF and 5 AMF deployments.

Scenario 2 – Incremental UE Load: UE count started at 100 and increased by 10 UEs every 5 seconds until reaching 200 UEs. This scenario evaluates the framework's responsiveness and registration time under progressively increasing load.

(a) Registration time with different numbers of UEs



(b) Registration time with incremental numbers of UEs

Fig. 3: Registration time

C. Implementation Notes

In the experimental setup, Open5GLoS maintains abstraction between the RAN and the control plane by acting as a proxy for NGAP control messages. Upon initialization, the gateway discovers active AMFs in the Kubernetes cluster and establishes SCTP associations via NGSetupRequest/Response procedures. When a UE initiates registration, the gateway intercepts the NGAP Initial UE Message, applies the NGAP-aware load-balancing algorithm, and forwards the message to the selected AMF after rewriting the UE identifiers. This design ensures transparency for the gNB and UEs, while enabling per-UE load balancing and dynamic AMF scaling.

D. Performance Metrics

The following metrics were measured in both scenarios:

Registration Time: Elapsed time from the UE's 5GMM-Deregistered.PLMN-SEARCH state to receiving Registration Accept.

Registration Success Rate: Percentage of UEs that successfully completed registration with an AMF.

E. Results and Analysis

Figures 3a and 3b illustrate the variation in UE registration time for the two scenarios under different AMF configurations.

TABLE I: Performance comparison of 1 AMF and 5 AMFs

Scenario	Config	Avg. Time [s]	Success [%]
Fixed (100 UEs)	1 AMF	7.7	100
	5 AMFs	7.2	100
Fixed (150 UEs)	1 AMF	5.3	100
	5 AMFs	4.9	100
Fixed (200 UEs)	1 AMF	5.4	97
	5 AMFs	4.6	100
Fixed (250 UEs)	1 AMF	–	0
	5 AMFs	4.1	100
Incremental (max 200 UEs)	1 AMF	6.1	88
	5 AMFs	4.7	100

In both scenarios, the 5 AMF deployment consistently shows shorter registration times compared to the single-AMF setup. In Scenario 1 (fixed UE load), registration time remains relatively low for the 5 AMF case even as the number of UEs increases, whereas the single-AMF configuration experiences a sharp increase and fails to register any UEs at 250. In Scenario 2 (incremental UE load), the registration time in the 5 AMF deployment increases more gradually, indicating better scalability under rising traffic.

Based on these measurements, Table I summarizes the average registration times together with the calculated improvement percentages and registration success rates. For 200 UEs in Scenario 1, the 5 AMF deployment reduces registration time by approximately 18% compared to the single-AMF case, while in Scenario 2 the improvement reaches about 36.9%. Furthermore, the 5 AMF configuration achieves a 100% success rate in all test cases, including scenarios where the single-AMF deployment fails.

V. Conclusion

The proposed Open5GLoS framework demonstrates that open-source 5G Core systems can achieve reliable, cloud-native AMF scaling with NGAP-aware load balancing while maintaining UE session continuity. Beyond performance gains, the results reveal critical gaps in current 3GPP specifications—particularly in identity management, service discovery, and transport-layer load balancing—that hinder native support for elasticity at scale. These insights suggest that standardization bodies should adapt NF designs to align with cloud-native principles from the outset. While the current work focuses on AMF, similar approaches could be extended to SMF, UPF, and other control-plane functions. Future research will explore predictive and AI-driven scaling mechanisms and evaluate their integration into multi-vendor, heterogeneous environments, contributing both practical tools for deployment and informed recommendations for evolving 5G/6G architectures. Open5GLoS is publicly available to support reproducibility and foster further research.

VI. Acknowledgement

This work was supported by the ICT R&D program of MSICT/IITP. [RS-2024-00405354, Development of Evolved SBA Framework and Core Technologies of Control/User Plane NFs]

References

[1] 3GPP, "5G: Study on scenarios and requirements for next generation, Technical Report (TR) 138.913," 3GPP, Tech. Rep., April. 2022, v.17.0.0.

[2] ——, "System architecture for the 5G system (5GS), Technical Specification (TS) 123.501," 3GPP, Tech. Rep., July. 2022, v.17.0.5.

[3] D. Scotece, A. Noor, L. Foschini, and A. Corradi, "5g-kube: Complex telco core infrastructure deployment made low-cost," *IEEE Communications Magazine*, vol. 61, no. 7, pp. 26–30, 2023.

[4] Kubernetes, https://kubernetes.io/, 2014, accessed: 2025-07-23.

[5] 3GPP, "Procedures for the 5g system (5GS), Technical Specification (TS) 23.502," 3GPP, Tech. Rep., Sep. 2024, v.17.13.0.

[6] ——, "5G - NG-RAN: Ng application protocol (ngap), Technical Specification (TS) 38.413," 3GPP, Tech. Rep., Oct. 2021, v.16.7.0.

[7] Free5gc, https://free5gc.org/, 2019, accessed: 2025-07-23.

[8] Open5GS, https://open5gs.org/, 2017, accessed: 2025-07-23.

[9] V.-G. Nguyen, K.-J. Grinnemo, J. Taheri, and A. Brunstrom, "On load balancing for a virtual and distributed mme in the 5g core," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1–7.

[10] A. Chouman, D. M. Manias, and A. Shami, "A reliable amf scaling and load balancing framework for 5g core networks," in *2023 International Wireless Communications and Mobile Computing (IWCMC)*, 2023, pp. 252–257.

[11] T. V. K. Buyakar, A. K. Rangisetti, A. A. Franklin, and B. R. Tamma, "Auto scaling of data plane vnfs in 5g networks," in *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–4.

[12] Y. Bello, A. A. Abdellatif, M. S. Allahham, A. R. Hussein, A. Erbad, A. Mohamed, and M. Guizani, "B5g: Predictive container auto-scaling for cellular evolved packet core," *IEEE Access*, vol. 9, pp. 158 204–158 214, 2021.

[13] V.-G. Nguyen, K.-J. Grinnemo, J. Taheri, and A. Brunstrom, "Adaptive and latency-aware load balancing for control plane traffic in the 4g/5g core," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2021, pp. 365–370.

[14] W. Dev, S. Bera, and A. Tiwari, "Control-plane load balancing and auto-scaling in 5g and beyond networks," in *2025 International Conference on Information Networking (ICOIN)*, 2025, pp. 83–87.

[15] LoxiLB, https://github.com/loxilb-io/loxilb, 2020, accessed: 2025-07-23.

[16] UERANSIM, https://github.com/aligungr/UERANSIM, 2020, accessed: 2025-07-23.

[17] Q.-H. Tran, "5glos," https://github.com/HasukiHT/5glos, 2025, accessed: 2025-07-23.