Prototyping and Load Balancing the Service Based Architecture of 5G Core using NFV

Tulja Vamshi Kiran Buyakar
Department of CSE
IIT Hyderabad, India
cs16mtech11020@iith.ac.in

Harsh Agarwal
Department of CSE
IIT Hyderabad, India
cs15btech11019@iith.ac.in

Bheemarjuna Reddy Tamma
Department of CSE
IIT Hyderabad, India
tbr@iith.ac.in

Antony Franklin A
Department of CSE
IIT Hyderabad, India
antony.franklin@iith.ac.in

***Abstract*—3GPP has chosen Service Based Architecture (SBA) for 5G Core (5GC). In this work, we build a prototype of SBA of 5GC from scratch using open source tools in Network Functions Virtualization (NFV) environment. In SBA, Network Functions (NFs) are designed to expose capabilities to consumers with interfaces, called Service Based Interface (SBI). To reduce the latency and the load on NFs, we use gRPC, a modern open-source Remote Procedure Call (RPC) framework, instead of REST for implementing SBI. We present a distributed service registration and discovery framework for 5GC using a software solution named Consul. We then propose usage of a Look Aside Load Balancer (LALB) for load balancing multiple NFs of 5GC. The control plane latency is reduced by routing traffic across multiple instances of Access and Mobility Management Function (AMF) via an LALB. Experimental results suggest that carefully chosen load balancing algorithms can significantly lessen the control plane latency when compared to simple random or round-robin schemes.**

I. Introduction

To support the diverse requirements of 5G, 3GPP has proposed two architectural representations for 5G Core in Release 15 [1] - a full reference point representation and a service-based representation. As the standards progressed [2], Service Based Architecture (SBA) was selected as the architecture going forward.

Communication among the Network Functions (NFs) leverage HTTP based APIs, replacing protocols like Diameter. The use of HTTP based APIs is a turning point in the 5G core since it reduces dependencies between functions that multiply in 5GC [3]. The NFs are designed to expose capabilities to consumers with interfaces, called Service Based Interface (SBI). Using an SBI makes it faster to add and remove instances from service paths in 5GC. Using technologies like Software Defined Networking (SDN) and Network Functions Virtualization (NFV) in cloud infrastructure makes it easy to deploy 5GC and also reduces Capital Expenditure (CAPEX) and Operating Expenditure (OPEX).

The heavy bursts of signaling traffic in the 5GC require it to be robust. Hence, it is necessary to implement a highly scalable and resilient architecture of the control plane that can dynamically respond to any changes in the network, e.g., the number of connected devices. Having a pool of NFs requires a load balancer in front of them to distribute the incoming traffic among them. But we cannot utilize the scaled NF instances to their full potential if the load balancer becomes a bottleneck.

In this work, we implement the SBA of 5G Core from scratch and deploy it in an NFV environment. To reduce the latency and the load on the NFs, we make use of Google Remote Procedure Call (gRPC) [4]a modern open-source Remote Procedure Call (RPC) framework, instead of HTTP REST as the SBI. We implement a distributed setup for the Network Function Repository Function (NRF) for service registry and service discovery, using Consul [5], an open-source distributed and highly available service discovery and configuration system. We propose using a look-aside load balancer [6] (LALB) instead of a proxy-based load balancer to meet the high scalability and low latency requirements of the 5G control plane.

II. Related Work

The authors in [7] and [8] discuss the motivation for SBA and related technologies which can be used to realize SBA. To realize SBA, the authors in [9] proposed Service Level Virtualization (SLV) which decomposes each network entity into various service blocks to provide respective services. In [10], the authors presented the state-of-art methods in service discovery with reflections on the specific needs of microservice architecture and in particular in the context of telecom applications. The authors in [11] conclude that NFV based implementation is better suited for networks with high signaling traffic.

In [12], the authors virtualized the Mobility Management Entity (MME) of 4G core by applying NFV principles and then deployed it as a cluster of multiple virtual MME instances (vMMEs) with a front-end load balancer. Similarly, the authors in [13] proposed a cloud native MME architecture using an L7-Load Balancer for packet inspection and forwarding to the respective entity in the MME VNF pool.

Most of the works as mentioned above explain the design and choices for 5G Core realization. However, they do not prototype the concept of a full-fledged 5G Core involving various components of it. In this work, we focus on building a full prototype including the protocol stack of SBI for 5G, with multiple instances of each NF and a load-balancer to handle the bulk load. Our main contributions in this work are:

- Prototyping SBA of 5G Core using gRPC as SBI in an NFV environment.

- Prototyping a distributed architecture for service registry and discovery in the NRF using Consul.
- Proposing LALB based design for load balancing NFs in 5G Core.

III. Realization of 5G-SBA

For the realization of SBI, the authors in [8] have proposed and evaluated various protocol stacks that are shown in Table I. We have therefore chosen gRPC based API design style which

TABLE I
REALIZATION OF SBI

Layer	Supported Protocols
Application Layer:	HTTP 1.1; HTTP 2.0
Serialization and Deserilization:	JSON; BSON; ProtoBuf
Transport Layer:	TCP; UDP; QUIC
API Design:	REST; RPC

uses HTTP 2.0 as Application Layer Protocol and ProtoBuf for Serialization and Deserialization. Our choice for SBI is made clear in the following subsection.

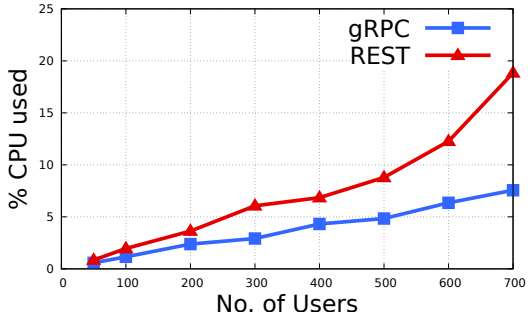

Fig. 1. Comparison of CPU utilization of gRPC vs REST.

A. Motivation for gRPC

In this work, we have used gRPC instead of REST. We run various experiments to know which of these two performs better concerning CPU utilization and latencies.

Benchmarking setup of gRPC and REST: A gRPC server is set up which accepts a client's payload as a Protocol Buffer message and relays it back to the client. An HTTP server is also set up which accepts the client's payload as JSON and relays it back to the client. In Fig. 1, we have plotted average CPU load on the gRPC and HTTP servers while increasing the number of clients. In Fig. 2, we have plotted average time taken to respond to a request by the gRPC and HTTP servers while increasing the number of clients.

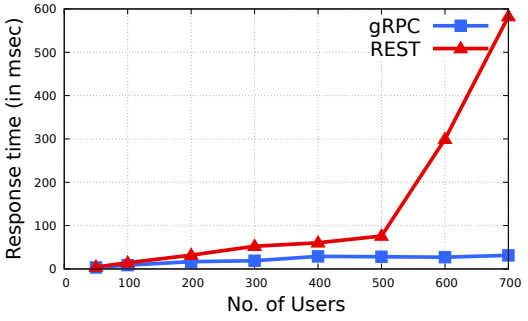

Fig. 2. Comparison of Avg. response time of gRPC vs REST.

It is observed from the figures that the CPU utilization and average response time for gRPC are lesser than that of REST. When compared to Protocol Buffers, unmarshalling JSON is a computationally expensive task which is resulting in higher values of CPU utilization and average response time. Also, gRPC works on HTTP 2.0 protocol, unlike most REST libraries which support only HTTP 1.1. Since HTTP 2.0 is multiplexed and has header compression, it is more efficient than HTTP 1.1. Hence, we have chosen gRPC for implementing SBI in our setup.

B. gRPC based 5G Core Architecture

Fig. 3 shows how our gRPC based 5GC architecture (gRPC-5GC) fits in ETSI NFV architecture [14]. The virtualization of resources in the architecture is done with the help of Docker platform [15]. The NFV Orchestrator (NFVO) interacts with Operations Support System / Business Support System (OSS/BSS) for VNF lifecycle management and policy management. While the VNF Manager (VNFM) interacts with Docker Engine to spawn new instances of the 5GC, the Virtual Infrastructure Infrastructure (VIM) interacts with underlying NFV Infrastructure (NFVI) for allocation of resources to the VNFs.

The gRPC-5G-Core is designed and implemented as per the standard 3GPP protocol stacks for AMF, AUSF, SMF, UPF including a RAN simulator and a Sink node. All the NFs are built as software modules in C++11 and run on individual Docker containers on a private cloud. The purpose of Radio Access Network (RAN) simulator is to generate uplink and downlink traffic to the 5GC. It produces multiple threads that simulate UEs which lead to control plane and data plane transmissions within the 5GC. The Sink node module is used to serve as a Packet Data Network (PDN) server to receive the generated uplink traffic and send back the acknowledgments as downlink traffic. The modules of AMF, AUSF, and SMF are run on multi-threaded servers to service the requests from other 5GC components and send the responses back. The AUSF simulates the behavior of HSS in LTE-EPC and uses a MySQL database for storing the details of various users. In this prototype, we have two different types of links. One is reference point based link, and other is service based link. Interfaces N2 and N4 follow the multi-threaded Stream Control Transmission Protocol (SCTP) architecture, whereas the N3 and N6 follow multi-threaded UDP architecture. N2, N3, N4, and N6 in Fig. 3 are the reference point based links. Service based links follow the multi-threaded gRPC architecture. NAmf, NSmf, NAusf, and NNrf in Fig. 3 are the service based links. After finishing the 5G system (5GS) bearer setup, each UE does some data transfer to the Sink. We have implemented the following 5G procedures in our prototype.

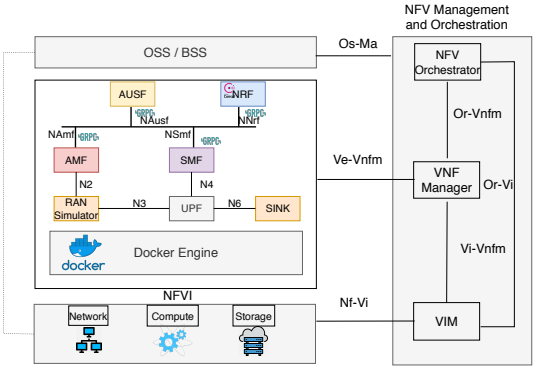

Fig. 3. gRPC based 5G Core in ETSI NFV Framework.

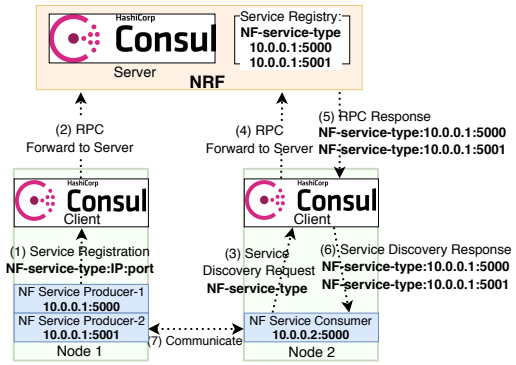

Fig. 4. NRF Service Registration & Discovery Setup.

- *UE Registration*: Message flow happens among RAN Simulator, AMF, AUSF & UDM involving the UE Authentication & Security Setup.
- *Session Creation*: For session creation, the call flow happens among AMF, SMF & UPF. After that, Bearer ID & UE_IP are sent to the RAN Simulator by the AMF. At the end of session creation, UEs are ready to do data transfer.
- *UE Uplink / Downlink Data Transfer*: An OpenVPN tunnel is set up between the RAN Simulator & Sink and the data flow is generated synthetically using iperf3 [16] which pumps TCP traffic from the RAN simulator.
- *UE Detach*: Consists of Session Deletion using Bearer ID & Tracking Area Identifier (TAI) values.

C. Service Registration and Discovery

In the SBA, whenever an NF needs to service a request, it contacts other NFs. Since it does not know where the other NFs are located, it initiates the Service Discovery procedure by communicating with the NRF. As NRF maintains the details of all the NFs, it responds with the appropriate NF.

The concept of service registration and discovery at NRF is implemented using Consul as shown in Fig. 4. The NRF runs as a Consul server on a dedicated server node. The NF service producers and consumers are on separate server nodes with every node running a Consul client.

When a new NF Service Producer is spawned, it registers itself with Consul by sending its type-of-service, IP Address, and port number to the Consul client running on the node. When an NF Service Consumer wants to communicate with other NF, it sends a service discovery request containing the type-of-service to the Consul client running on the node. The Consul Server retrieves the details of all the healthy instances of the requested type and returns them to the client, which then forwards it to the consumer. Although this work runs all the nodes in a single data center, it can easily be extended to multiple data centers by creating a federated Consul cluster.

D. Evaluation of Testbed

We evaluate our testbed by running multiple number of UE threads at RAN simulator. The network functions of gRPC-based-5GC are deployed on Intel Xeon CPU E5-2690 machine which has 54 cores with 64 GB RAM and 2 TB HDD, running the Ubuntu 16.04.5 LTS Operating System.

The testbed is evaluated with respect to control plane latencies incurred by the UEs, CPU utilization of various 5GC components, and average individual UE throughputs.

We define Control Plane Latency (CP_l) as the sum of various procedures of attach procedure latency ($attach_l$) (which is the sum of UE Registration and Session Creation latencies) and detach procedure latency ($detach_l$).

$$CP_l = attach_l + detach_l \tag{1}$$

In this experiment, only a single instance of each NF of 5GS is considered for processing the requests. There is no limit on the number of cores an NF can use. With the help of the Docker platform, NF can increase utilization of CPU cores based on processing requirement.

1) CP_l Measurement: From Fig. 5, we observe that CP_l increases on increasing the number of UEs. To reduce CP_l with the increasing number of UEs, we need multiple instances of each NF and a load balancer to distribute traffic load among them. The next section discusses more on load balancing.

2) CPU Utilization Measurement: While we run various concurrent UE threads at RAN simulator, we measure the CPU utilization at AMF, SMF, and AUSF. As the processing load increases on the NFs, they utilize more number of CPU cores. Hence we see the CPU utilization of NFs going higher than 100%. From Fig. 6, we can observe that CPU utilization increases on increasing the number of UEs. Among all the NFs, the AMF's CPU utilization is very high since it handles a lot of signaling messages. If multiple AMFs are not used, then it may lead to session failures.

3) Dataplane Throughput Measurement: The network bandwidth is shared across various concurrent UEs that are transferring data. iperf3 equally distributes the available bandwidth across all currently running iperf sessions. As a result, we can see a reduction in individual UE throughputs with an increase in the number of UEs in Fig. 7.

IV. Load-Balancing Architecture for GRPC based 5G Core

From Figs. 5 & 6, we conclude that for a single instance of each NF handling a large number of UEs, CP_l and CPU utilization of 5GC components increase drastically.

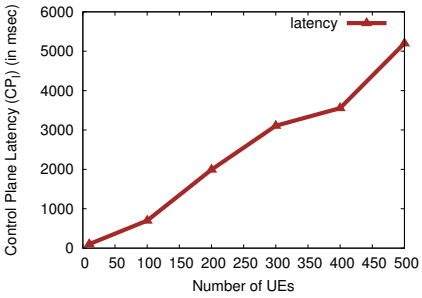

Fig. 5. Variation of CP_l with Concurrent UEs.

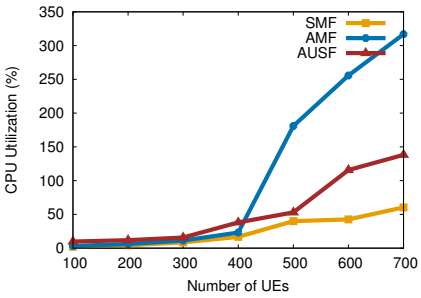

Fig. 6. CPU Utilization of various NFs.

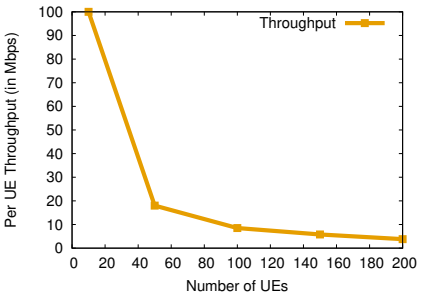

Fig. 7. Per UE throughput with Concurrent UEs.

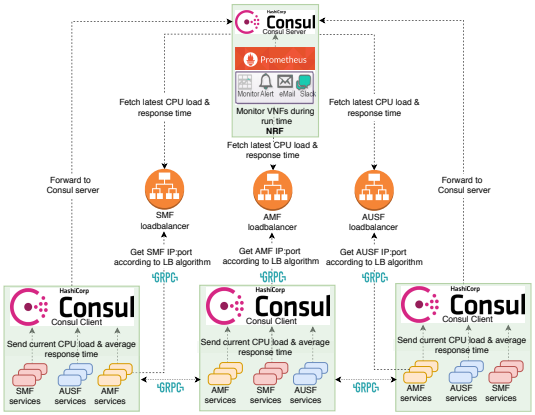

Fig. 8. Implementation Framework of LALB.

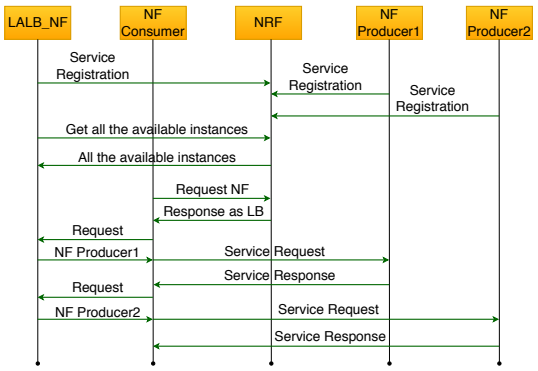

Fig. 9. Load balancer interaction with NRF.

If there are multiple instances of each NF, many UEs can be handled concurrently and hence leads to reduction in the CP_l. To properly route the traffic, we need a Load Balancer (LB) which communicates with multiple instances of the respective NF and routes the traffic among those instances.

A. Look aside load balancing

In general, there are two types of load balancing architectures: proxy and client-side. In proxy load balancing, the client issues RPC to an LB proxy, which distributes it to one of the available backend servers. Although this is simple to implement, this has a higher latency because the LB is in the data path.

In client-side load balancing, the client is aware of the multiple backend servers and chooses one of the servers to use for each RPC. Although this has lower latency because of the elimination of an extra hop, this adds to the complexity of the client.

Hence, we use a variant of client-side load balancing, called *look aside load balancing* [6]. There is a special LB server called the Look Aside Load Balancer (LALB). The client queries the LALB, and the LALB responds with the details of the best server to use. The client then directly interacts with the backend server. The servers share their load reports with the LALBs. This approach has low latency because there is no extra hop in the data path. The strategy is also scalable because to scale the LALBs horizontally just the load report needs to be shared with the new instances.

B. Implementation Framework

The implementation framework of LALB has been shown in Fig. 8. The implementation uses Consul along with Docker container platform for virtualizing the NFs. Prometheus [17] is used as a monitoring tool to monitor the NFs.

The NRF node runs a Consul server and a Prometheus server on top of it. The other nodes run a *Consul Client* and the 5G NFs in Docker containers. There are 3 LALBs, one each for AMF, AUSF, and SMF. Every NF instance of a 5G component initially registers its IP address and port number and the type of service it offers with Consul. The NF periodically records its CPU utilization and mean response time and updates them in the service registry on the Consul server in the NRF.

The LALBs periodically query the NRF for fetching the details of all available NF producer instances. Fig. 9 shows the call flow diagram of the Load balancer interaction with NRF and other NFs.

When an NF Consumer wants to access a service, it requests details of the load-balancer handling that particular service from the NRF. It then contacts that load balancer which responds with one of NF producers depending on the load balancing scheme. The NF consumer then directly communicates with the NF producer to access the service.

V. Evaluation of AMF LALB

In this section, we evaluate LALB on how effectively it can handle a large number of requests. For the evaluation,

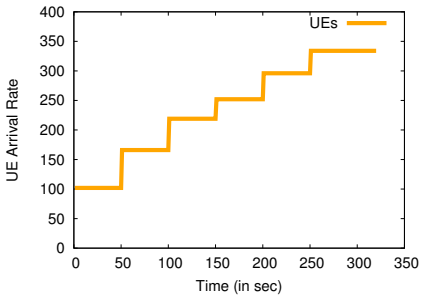

Fig. 10. Average UE Arrival Rate with time.

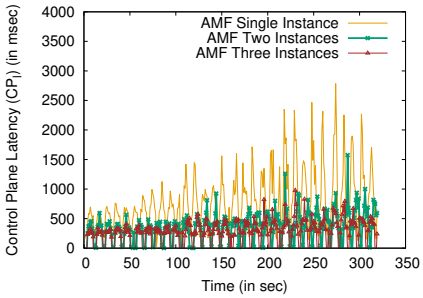

Fig. 11. CP_l with time.

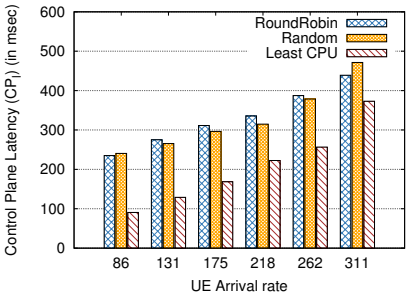

Fig. 12. CP_l with Avg. UE Arrival Rate.

we are considering the load balancing of AMF because from Fig. 6, we can observe that it is the most computationally expensive NF. In this experiment, unlike our previous test, we limit the number of cores per NF. Our experimental setup runs the 5G NFs as Docker containers on top of commodity hardware servers. The evaluation of LALB is done in two ways:

1) Reduction of CP_l by increasing number of instances: The average rate of UEs that are arriving per second is shown in Fig. 10. Fig. 11 presents the variation of CP_l with single and multiple instances of AMF. Round-Robin policy is used for load-balancing among the multiple instances of AMF. With multiple instances of AMF, it is observed that there is much reduction in CP_l when compared to a single instance. This difference is mainly due to high concurrency rate provided by the multiple instances of AMF.

2) Variation of CP_l with various load balancing schemes: CP_l can be reduced further with the help of relevant load balancing schemes. In this section, we have evaluated the following load balancing mechanisms with respect to CP_l of the individual AMF instances.

- *Round-Robin (RR)*: In this mechanism, the load balancer follows a round-robin approach on a list of servers in the group.
- *Random (RD)*: The load balancer forwards the client's request to a randomly chosen server from the list of servers.
- *Least CPU based (LCU)*: In this mechanism, the load balancer always chooses the server whose current CPU utilization is the least. In this way, this mechanism can have high concurrency rate, since no server in the list is underutilized.

In Fig. 12, we see that CP_l for RR and RD is almost same. This is because, in RR and RD, all AMF instances are equally loaded with respect to CPU utilization. Hence, all the requests face similar contention in all the AMF instances. Fig. 12 shows that CP_l for LCU is lesser than both RR and RD because in LCU the consumer accesses AMF instance that is currently least loaded AMF. Hence the consumer's request faces very less contention in the AMF and is processed at a much faster rate. Therefore picking an appropriate load-balancing policy plays a vital role in building a scalable SBA for 5GC. Improper load-balancing schemes can lead to underutilization of scaled NF instances and thereby could lead to higher CP_l.

VI. CONCLUSION

In this paper, we have presented a model for the realization of SBI and SBA for 5GC and evaluated it with respect to the control plane latency and resource utilization of various 5GC components and data plane throughput. We have proposed an LALB based design for load balancing of UE load across multiple NFs in 5GC, thereby decreasing the control plane latency with multiple instances and load balancers.

ACKNOWLEDGEMENT

This work is supported by the R&D work undertaken in the project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology (MeitY), Govt. of India, being implemented by Digital India Corporation and "Converged Cloud Communication Technologies" of MeitY, Govt. of India.

REFERENCES

[1] 3GPP, "3GPP TS 23.501 - System Architecture for the 5G System," 2018.
[2] 3GPP, "3GPP TS 23.502 - Procedures for the 5G System," 2018.
[3] Huawei, "Service Based Architecture for 5G Core Networks," https://ubm.io/2BZISmW, 2018.
[4] Google, "gRPC," https://grpc.io/, 2018.
[5] HashiCorp, "Consul," https://consul.io/, 2018.
[6] "Lookaside load balancing in gRPC," https://grpc.io/blog/loadbalancing, 2018.
[7] NGMN, "Service Based Architecture in 5G ," https://bit.ly/2GOg9qP, 2018.
[8] C. Zhang, X. Wen, L. Wang, Z. Lu, and L. Ma, "Performance Evaluation of Candidate Protocol Stack for Service-Based Interfaces in 5G Core Network," in *IEEE ICC Workshops*, 2018.
[9] B.-J. Qiu, Y.-S. Hsueh, J.-C. Chen, J.-R. Li, Y.-M. Lin, P.-F. Ho, and T.-J. Tan, "Service Level Virtualization (SLV): A Preliminary Implementation of 3GPP Service Based Architecture (SBA)," in *ACM MobiCom*, 2018.
[10] C. Rotter, J. Illés, G. Nyíri, L. Farkas, G. Csatári, and G. Huszty, "Telecom strategies for service discovery in microservice environments," in *ICIN*, 2017.
[11] A. Jain, N. Sadagopan, S. K. Lohani, and M. Vutukuru, "A comparison of SDN and NFV for re-designing the LTE packet core," in *IEEE NFV-SDN*, 2016.
[12] 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 *IEEE PIMRC*, 2018.
[13] P. 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 *IEEE PIMRC*, 2017.
[14] ETSI, "ETSI NFV Architecture," https://www.etsi.org/technologies-clusters/technologies/nfv, 2017.
[15] Docker, "Docker," https://www.docker.com/, 2018.
[16] "iperf3," https://iperf.fr/, 2017.
[17] "Prometheus," https://prometheus.io/, 2018.