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 latencies are 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

The fifth generation (5G) of mobile technology dramatically improves user experienced speeds and handles enormous amounts of network traffic. Massive amounts of data can be transferred instantly in 5G networks with higher data rates and low latency. It will be the foundation for applications like virtual reality, autonomous driving, and the Internet of Things (IoT). Since 5G serves a lot of use cases, it has diversified requirements for these use cases.

The International Telecommunication Union (ITU), classified 5G use cases into three broad families, namely enhanced Mobile Broadband (eMBB), ultra-Reliable Low-Latency Communications (uRLLC), and massive Machine Type Communication (mMTC) [1]. The eMBB services are required for applications which require high bandwidth and sustained high capacity network connections, such as High Definition (HD) videos, Virtual Reality (VR) and Augmented Reality (AR). The uRLLC services are required for applications which have latency constraints and require high reliability and availability like Factory Automation, Tactile Internet, Intelligent Transportation Systems, Process Automation, etc. The mMTC focuses on emerging smart services like health care, manufacturing, utilities, consumer goods, smart agriculture, smart city, etc. which require the large-scale and dense deployment of devices.

The fundamental requirements of a 5G network according to [2] are:

- support for various use cases like eMBB, mMTC and uRLLC adhering to their diverse requirements.
- support for easy deployment and maintenance. Each functionality should be able to upgrade according to new requirements.
- support for emerging new services, which require exposing network capabilities to operator's services and third-party applications.

To support the above-mentioned requirements, 3GPP has proposed two network architectures for 5G Core in Release 15 [3] - a full reference point representation and a service-based representation. As the standards progressed [4], Service Based Architecture (SBA) was selected as the architecture going forward. The SBA separates control plane and user plane unlike in 4G Core, which enables independent scaling and deployment of Network Functions (NFs). SBA representation of the 5G Core Network (5GC) architecture enables the NFs to authorize other NFs and access their services.

Communication among these NFs leverage HTTP based APIs, replacing protocols like Diameter [5]. The use of HTTP based APIs is a turning point in the mobile core since it reduces dependencies between functions that multiply in 5GC [6]. The NFs are designed to expose capabilities to consumers with interfaces, called Service Based Interface (SBI). SBI should be light-weight for efficient communication and be easy to expose internally and externally for invocation or reuse. Using an SBI makes it faster to add and remove instances from service paths in 5GC. The deployment of 5GC with technologies like Software Defined Networking (SDN) and Network Functions Virtualization (NFV) using cloud infrastructure makes it easy to deploy and also reduce 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 use Google Remote

Procedure Call (gRPC) [7], 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 [8], an open-source distributed and highly available service discovery and configuration system.

We propose using a look-aside load balancer [9] instead of a proxy based load balancer to meet the high scalability and low latency requirements of the 5G control plane. In a Look Aside Load Balancer (LALB), the clients query an LALB which responds with the best NF instance to use. The client then directly interacts with the NF instance. We evaluate our setup by measuring the control plane latency by varying User Equipment (UE) traffic load with different load balancing strategies. We intend to open-source this work for research purposes.

The rest of the paper is organized as follows. Section II presents an overview of related works. Section III describes SBA of 5GC as defined by 3GPP. Section IV describes our implementation of SBI using gRPC. Section V describes our implementation of the various 5GC components and the service registration and discovery setup using Consul and evaluates them on a testbed. Section VI proposes a load balancing architecture for 5GC. Section VII evaluates the proposed load balancer architecture for AMF on a testbed. Finally, Section VIII concludes the paper and outlines future work.

II. Related Work

The authors in [2] discussed the motivation for SBA and related technologies which could be used to realize SBA. They discussed the conceptual idea of service framework which included service registration, service authorization, and discovery. They put forth a few options on how SBA can be deployed with NFV.

The authors in [10] listed out various ways of implementing the protocol stack of SBI for 5G architecture. They also compared candidate solutions in terms of their characteristics and performance.

To realize SBA, the authors in [11] proposed Service Level Virtualization (SLV) which decomposes each network entity into various service blocks to provide respective services. Besides, they virtualized each service block independently into different virtual machines.

In [12], 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. In this context, the authors investigated and compared different open-source frameworks.

In [13], the authors designed and evaluated two open-source implementations of LTE Evolved Packet Core (EPC): one based on SDN principles and the other based on NFV, and presented a performance comparison of both EPCs. They concluded that NFV based implementation was better suited for networks with high signaling traffic.

In [14], the authors virtualized the Mobility Management Entity (MME) by applying NFV principles and then deployed it as a cluster of multiple virtual MME instances (vMMEs) with a front-end load balancer. Experimental results in their work suggested that carefully selected load balancing algorithms can significantly reduce the control plane latency.

In [15], the authors 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 (attach, detach, location update).

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 Network Function Repository Function (NRF) using Consul.
- Proposing LALB based design for load balancing NFs in 5G Core.

III. Background

A. 5G Core Network Architecture

Service Based Architecture (SBA) was selected for realization of the 5GC network. In SBA, a service is an atomized capability in a 5G network, with the characteristics of high-cohesion, loose-coupling, and independent management from other services. Fig. 1 illustrates the 5GC network architecture based on SBA.

SBA consists of various components, some of which are similar to 4G EPC ones. The most relevant ones are defined below.

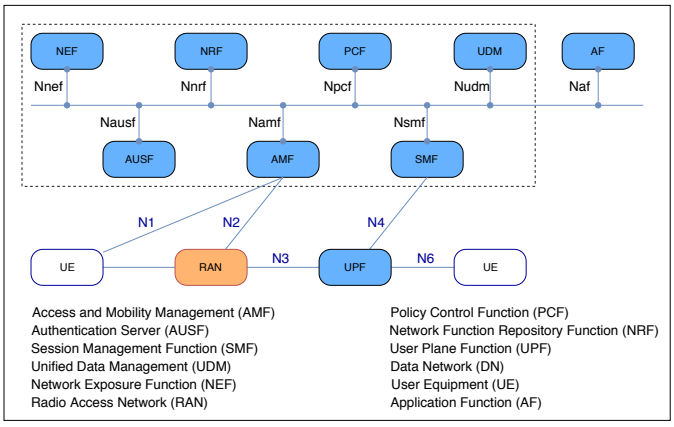

Fig. 1. Service Based Architecture for 5G Core.

- Access and Mobility Management Function (AMF): AMF handles access control and mobility. If fixed access is involved, mobility management will not be needed in

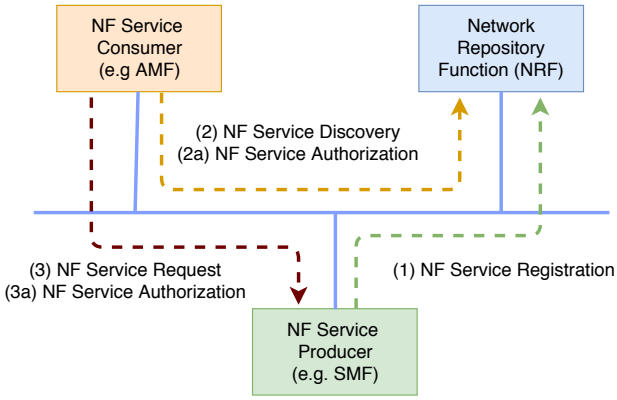

Fig. 2. Service Registration and Discovery with NRF.

the AMF. It is the entry point to the 5GC network, and improper management can lead to bottleneck issues.

- Authentication Server Function (AUSF): AUSF acts as an authentication server, which stores data for authentication of UE. It implements the Extensible Authentication Protocol (EAP) for authentication purposes.
- Session Management Function (SMF): SMF sets up and handles sessions according to operator-defined policies.
- Policy Control Function (PCF): PCF provides a policy framework for network slicing, roaming, and mobility management.
- User Plane Function (UPF): UPF handles and forwards users data traffic. It can be deployed according to the service type and in various configurations and locations.
- Unified Data Management (UDM): UDM stores subscriber data and profiles. It is similar to Home Subscriber Server (HSS) in LTE-EPC but might be used for both fixed and mobile access in 5G Core.
- Network Function Repository Function (NRF): NRF provides registration and discovery functionality so that the instances of network functions (NFs) can discover each other and communicate via APIs. The service registration and discovery procedures are followed as depicted in Fig. 2. When a new NF Service Producer (e.g., SMF) is spawned, it registers its details including the type-of-service, IP address, and port number with the NRF. When an NF Service Consumer (e.g., AMF) wants to communicate with the NF Service Producer (e.g., SMF), it sends a service discovery request to the NRF which first validates that the requesting instance is authorized to make such a request. If it is authorized, the NRF responds with the details of the registered SMF instance. The AMF then sends a service request to the SMF. The SMF then services the request after validating that the requesting instance is authorized to make the request.

B. gRPC

gRPC [7] is a modern, open source remote procedure call (RPC) framework widely used in micro-services based distributed systems. It provides features such as authentication, bidirectional streaming and flow control, blocking or non-blocking bindings, and cancellation and timeouts. It uses HTTP 2.0 for transport and Protocol Buffers as the interface description language. Protocol Buffers [16] is a method of serializing structured data. Protocol Buffer messages are strongly typed and are serialized into a compact binary-wire format. They are smaller, simpler, and much faster than JavaScript Object Notation (JSON).

C. Consul

Consul [8] is an open-source distributed and highly available service discovery, configuration, and segmentation system. Consul supports multiple data centers and provides a distributed key-value store and a secure way of doing service discovery and health checking. Every node that is part of the Consul cluster runs a Consul agent. A Consul server is where the data is stored and replicated. The servers participate in a consensus protocol to elect a leader among themselves and keep the data consistent among all the servers, across all data centers. A Consul client is a stateless entity that forwards all the requests to a Consul server, via an RPC.

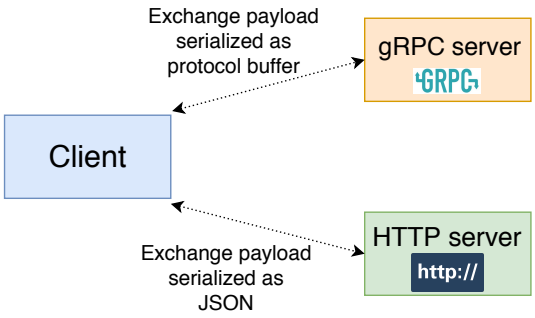

Fig. 3. Benchmarking setup of gRPC and HTTP REST.

IV. Realization of 5G-SBI

In the above section, we discussed the SBA of 5G Core. In this section, we will discuss the implementation of SBI which forms an important part in SBA. For the realization of SBI, the authors in [10] have proposed and evaluated various protocol stacks that are shown in Table I. In [10], the authors demonstrated HTTP 2.0 performs better than HTTP 1.1, and ProtoBuffers [16] is best suited for Serialization and Deserialization. We have therefore chosen gRPC based API design style which 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.

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

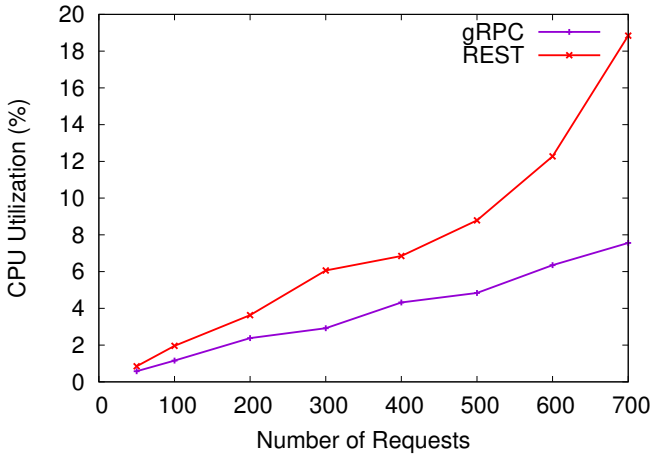

Fig. 4. Comparison of CPU utilization of gRPC vs REST.

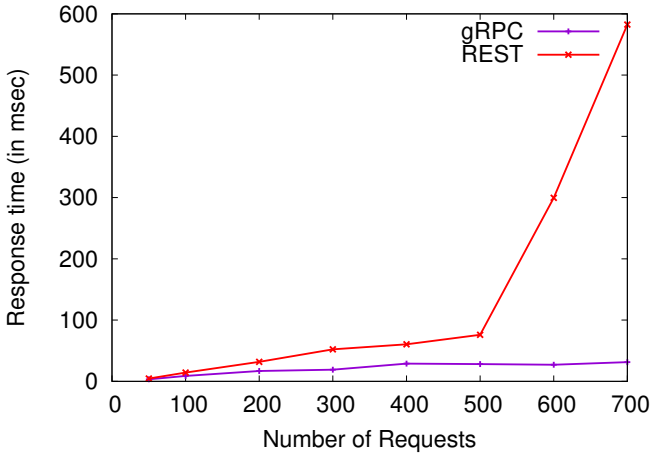

Fig. 5. Comparison of Avg. response time of gRPC vs REST.

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: The benchmarking setup is shown in Fig. 3. 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.

Client threads are set up which periodically query the two endpoints. The average response time taken to query a request and CPU utilization of the server to serve the requests are plotted by varying the number of clients. In Fig. 4, we have plotted average CPU load on the gRPC and HTTP servers while increasing the number of clients. In Fig. 5, we have plotted average time taken to respond to a request by the gRPC and HTTP servers while increasing the number of clients.

It is observed from the figures that the CPU utilization and average response time for gRPC is 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 use HTTP 1.1. HTTP 2.0 is multiplexed, it has header compression and is more efficient than HTTP 1.1. Hence, we have chosen gRPC for implementing SBI in our setup.

V. Realization of 5G-SBA

A. gRPC based 5G Core Architecture

Fig. 6 shows how our gRPC based 5G Core architecture (gRPC-5GC) fits in ETSI NFV architecture [17]. The virtualization of resources in the architecture is done with help of Docker platform [18]. 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. 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. The RAN simulator produces multiple threads that simulate UEs which lead to control plane and data plane transmissions with 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 acknowledgment as downlink traffic. The modules of AMF, AUSF, and SMF run multi-threaded servers which service the requests from other 5GC components and send the responses back. The AUSF simulates the behavior of HSS in LTE-EPC. The AUSF 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. 6 are the reference point based links. Service based ones follow the multi-threaded gRPC architecture. NAmf, NSmf, NAusf, and NNrf in Fig. 6 are the service based links.

Data Transfer: After finishing the 5G system (5GS) bearer setup, each UE does some data transfer to Sink. Data transfer from RAN to UPF happens with GTP The data transfer is implemented using iperf3 [19]. Sink implements the multi-threaded iperf server and listens on various ports. The data transfer happens via these iperf server threads. RAN simulator runs multi-threaded iperf clients that emulate UEs as the sender.

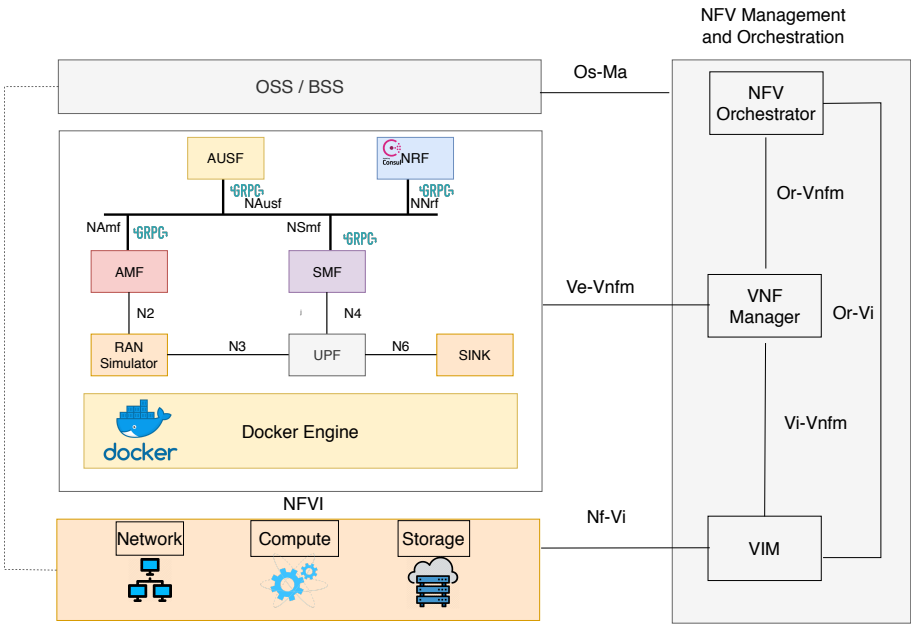

Fig. 6. gRPC based 5G Core in ETSI NFV Framework.

B. Service Registration and Discovery

In the SBA, whenever an NF needs to service a request, it contacts other NFs. But it does not know where the other NFs are located. So, 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. 7. 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

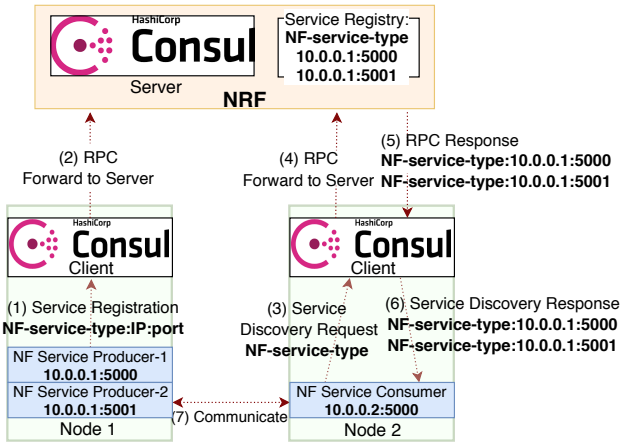

Fig. 7. NRF Service Registration & Discovery Setup.

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.

C. Call Flow among the NFs

Figs. 8 & 9 present the interaction of various NFs and the procedures involved. All the NFs initially do a Service Registration with NRF. We are not considering the radio interface between UE and RAN, as we are abstracting the UEs. The RAN simulator generates UE threads, and each UE thread sends an attach request along with IMSI number to AMF. After receiving the request, AMF proceeds to implement the various authentication and security procedures.

The AMF does a service discovery procedure with NRF to find the AUSF and SMF. It also caches this information for future reference. The AMF authenticates the UE with the help of AUSF. Then a mutual authentication takes place between the UE and the network. Various security setup procedures take place, during which encryption and integrity keys are exchanged between UE and AMF. AMF then contacts AUSF for the location update to check if AMF has changed since last detach. AUSF acknowledges the location update message by sending an update location acknowledgment which has IMSI and UE subscription data. Later, AMF contacts SMF for the session creation. The default bearer setup is finished by sending attach accept to RAN and receiving the attach complete from RAN. Afterward, the AMF undergoes the modify bearer procedure with SMF. Subsequently, the UE threads in the RAN simulator run data transfer through iperf on the default bearer. The UE session ends with a detach request to AMF, which contacts SMF for the detach procedure.

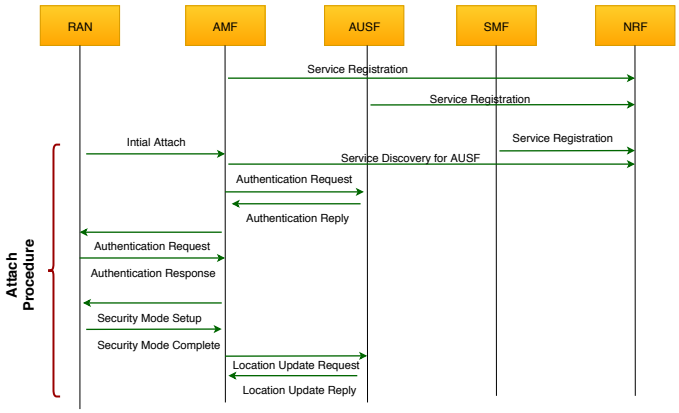

Fig. 8. Call Flow Diagram (1).

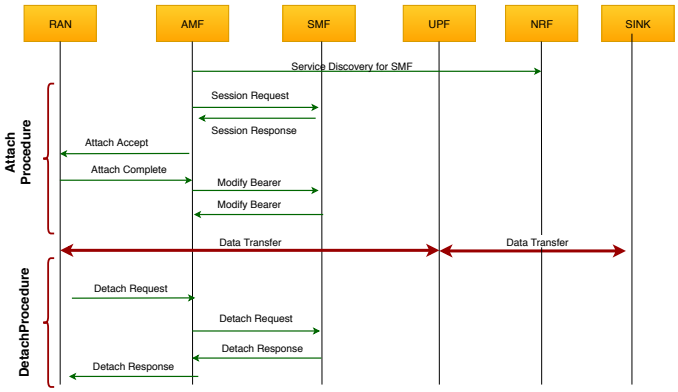

Fig. 9. Call Flow Diagram (2).

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 64GB RAM and 2 TB HDD, running the Ubuntu 16.04.2 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 sum of various procedures (shown in Fig. 8 and Fig. 9) of attach procedure latency ($attach_l$) and detach procedure latency ($detach_l$).

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

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

1) CP_l Measurement : From Fig. 10, 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 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 values of AMF, SMF, and AUSF. As the processing load increases on the NFs, they utilize a number of CPU cores. Hence we see the CPU utilization of NFs going higher than 100%. From Fig. 11, 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 5GS session failure.

3) Dataplane Throughput Measurement: Dataplane throughputs are measured using iperf3. 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. 12.

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

From Figs. 10 & 11, we conclude that for a single instance of each NF handling a large number of UEs, CP_l and CPU utilization of 5GC components increases drastically.

If there are multiple instances of each NF, many UEs can be handled concurrently and hence reduce the CP_l. To properly route the traffic, we need a Load Balancer (LB) which communicates with multiple instances of the NF and routes the traffic among those instances. To suit the SBA, we define an SBI for load balancer of each NF as shown in Fig. 13.

A. Look aside load balancing

In general, there are two types of load balancing topologies: 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. The backend servers share their load report with the LB, and the LB implements algorithms for allocating the load fairly.

In client side load balancing, the client is aware of the multiple backend servers and implements load balancing algorithms to choose one server to use for each RPC. Load report is shared with the client on the same connection on which the client RPC is executed.

Proxy load balancing is simple to implement and works with untrusted clients. But this has a higher latency because the LB is in the data path and its throughput may limit scalability.

Client side balancing gives high performance because of the elimination of an extra hop. But this adds to the complexity of the client and adds a maintenance burden. Clients must be trusted because they directly interact with the servers.

Hence we use a variant of client-side load balancing, called *look aside load balancing* [9]. There is a special LB server called the Look Aside Load Balancer (LALB). The client query the LALB, and the LALB implements load-balancing

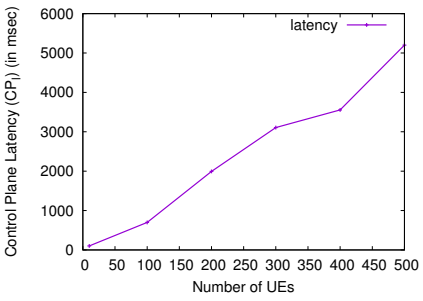

Fig. 10. Variation of CP_l with Concurrent UEs.

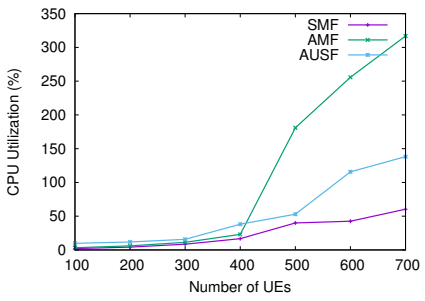

Fig. 11. CPU Utilization of various NFs.

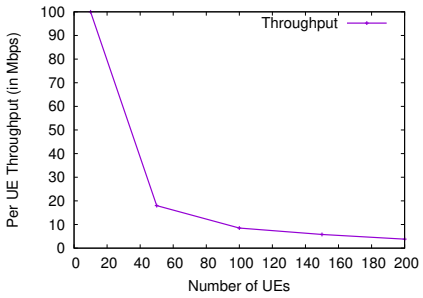

Fig. 12. Individual UE throughputs with Concurrent UEs.

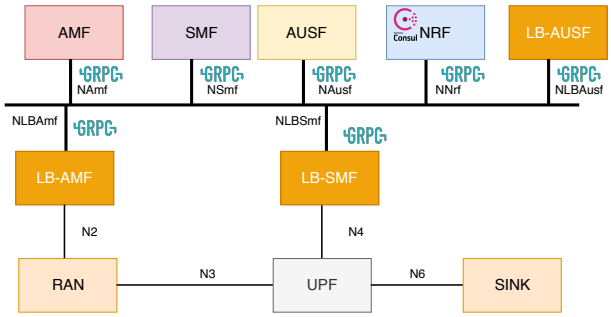

Fig. 13. Load balancer SBIs in SBA-5G.

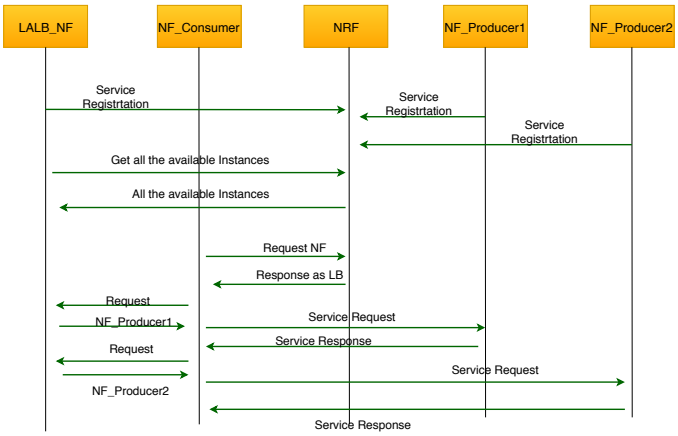

Fig. 14. Load balancer interaction with NRF.

algorithms to respond with the best server to use. The client then directly interacts with the backend server. The servers share their load reports with the LALBs. The clients can be untrusted because the trust boundary is handled by the LALB. 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 has been shown in Fig. 15. The implementation uses Consul along with Docker container platform for virtualizing the NFs. Prometheus [20] is used as a monitoring tool to monitor the NFs. Prometheus is an open-source monitoring and alerting toolkit.

The NRF node runs a Consul and a Prometheus server on top of it. The other nodes run a *Consul Client* and the 5G NFs 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 calculates 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. 14 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 the load-balancer details 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.

VII. Evaluation of AMF LALB

In this section, we evaluate LALB on how effectively it can handle large number of requests. For the evaluation we are considering the load balancing of AMF NF because from Fig. 11, we can observe that it is the most computationally expensive NF.

TABLE II
Configuration of Testbed.

Name	#Instances	#CPU Cores
AUSF	1	1
AMF	3	1
SMF	1	1
UPF	1	2
RAN	1	2
SINK	1	2
NRF	1	3

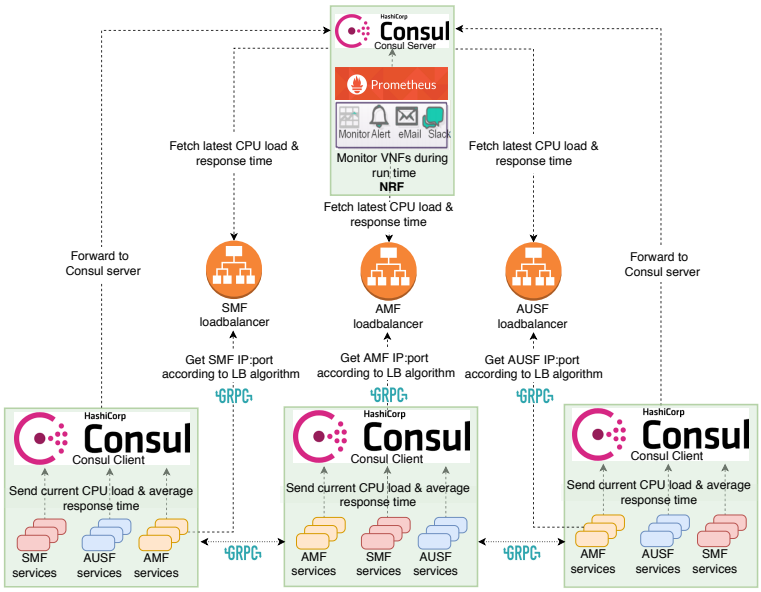

Fig. 15. Implementation Framework.

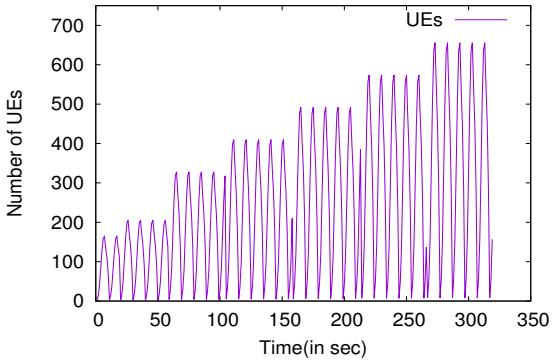

Fig. 16. Variation of UEs with time.

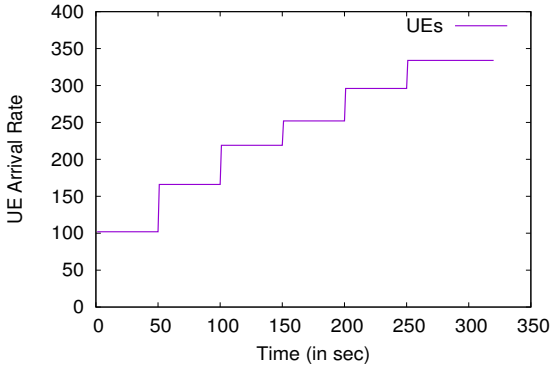

Fig. 17. Average UE Arrival Rate with time.

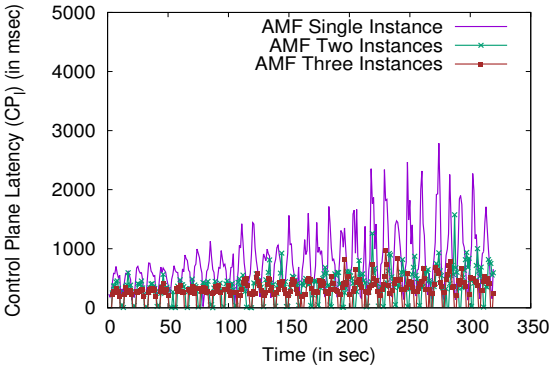

Fig. 18. CP_l with time.

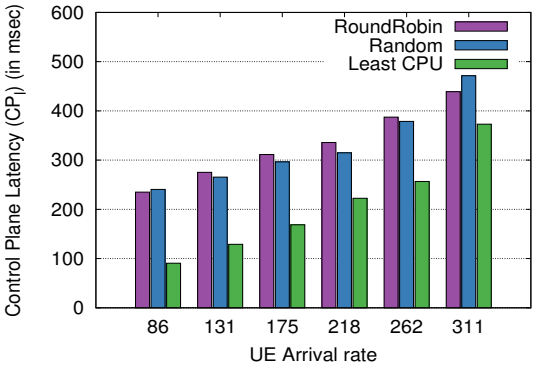

Fig. 19. CP_l with Avg. UE Arrival Rate.

In this experiment, unlike our previous test, we limit the number of cores per NF as given in Table II. 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:

- Measuring the reduction of CP_l by increasing the number

of AMF instances.
- Observing the variation of CP_l with various load balancing schemes.

1) Reduction of CP_l with increasing instances: Fig. 16 shows the variation of UE load on 5G Core. This variation follows a Poisson distribution for the UE arrival. The average rate of UEs that are arriving per second is shown in Fig. 17. Fig. 18 presents the variation of CP_l values with single and multiple instances of AMF. Round-Robin policy is used for load-balancing the multiple instances of AMF. With multiple instances of AMF, it is observed 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 chosen Round Robin, Random, and Least CPU based load balancing mechanisms and evaluated them 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 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. 19, 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 concerning CPU utilization. Hence, all the requests face similar contention in all the AMF instances. Fig. 19 shows that CP_l for LCU is lesser than both RR and RD because in LCU the consumer accesses the 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 lead to higher CP_l.

VIII. CONCLUSION AND FUTURE WORKS

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 Dataplane 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.
As future work, we intend to extend our work to realize the concept of Network Slicing.

REFERENCES

[1] I. T. Union, "Key features and requirements of 5G/IMT-2020 networks," https://bit.ly/2CedjWX, 2018.

[2] NGMN, "Service Based Architecture in 5G ," https://bit.ly/2GOg9qP, 2018.

[3] 3GPP, "3GPP TS 23.501 - System Architecture for the 5G System," 2018.

[4] 3GPP, "3GPP TS 23.502 - Procedures for the 5G System," 2018.

[5] "Diameter Protocol for LTE," https://bit.ly/2FbFQAh, 2017.

[6] Huawei, "Service Based Architecture for 5G Core Networks," https://ubm.io/2BZISmW, 2018.

[7] Google, "gRPC," https://grpc.io/, 2018.

[8] HashiCorp, "Consul," https://consul.io/, 2018.

[9] "Lookaside load balancing in gRPC," https://grpc.io/blog/loadbalancing, 2018.

[10] 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 International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[11] 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 *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 2018, pp. 669–671.

[12] C. Rotter, J. Illés, G. Nyíri, L. Farkas, G. Csatári, and G. Huszty, "Telecom strategies for service discovery in microservice environments," in *Innovations in Clouds, Internet and Networks (ICIN), 20th Conference on*. IEEE, 2017, pp. 214–218.

[13] A. Jain, N. Sadagopan, S. K. Lohani, and M. Vutukuru, "A comparison of SDN and NFV for re-designing the LTE packet core," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE, 2016, pp. 74–80.

[14] 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 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1–7.

[15] 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 *Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE 28th Annual International Symposium on*. IEEE, 2017, pp. 1–7.

[16] Google, "Protocol Buffers," https://developers.google.com/protocol-buffers/, 2018.

[17] ETSI, "ETSI NFV Architecture," https://www.etsi.org/technologies-clusters/technologies/nfv, 2017.

[18] Docker, "Docker," https://www.docker.com/, 2018.

[19] "iperf3," https://iperf.fr/, 2017.

[20] "Prometheus," https://prometheus.io/, 2018.