

Operational Aspects of Network Slicing in LTE and 5G Architectures

Tulja Vamshi Kiran Buyakar

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology



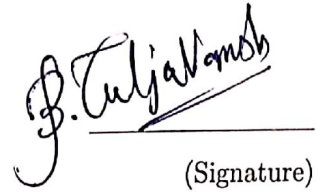
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Department of Computer Science and Engineering

June 2019

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

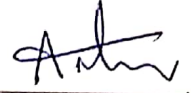

(Signature)

(Tulja Vamshi Kiran Buyakar)

CS16MTECH11020
(Roll No.)

Approval Sheet

This Thesis entitled Operational Aspects of Network Slicing in LTE and 5G Architectures by Tulja Vamshi Kiran Buyakar is approved for the degree of Master of Technology from IIT Hyderabad



(Dr. Antony Franklin A) Examiner
Dept. of CSE
IITH

Mr. Pandey
for Dr. Saurabh Joshi

(Dr. Saurabh Joshi) Examiner
Dept. of CSE
IITH



(Dr. Bheemarjuna Reddy Tamma) Adviser
Dept. of CSE
IITH

Mr. V. Pandey

(Dr. M.V. Panduranga Rao) Chairman
Dept. of CSE
IITH

Acknowledgements

I want to thank everyone who encouraged me during my M.Tech. Program at IIT Hyderabad. First and foremost, I must express my sincere gratitude to my advisor Dr. Bheemarjuna Reddy Tamma for accepting me as his student. He is one of the most hard-working and disciplined persons I have met in my life. His consistent mentoring and support to pursue my research interests and engagement through the learning process during the Master's program helped me to acquire many skills. I feel very graced to have worked with him and owe enormous gratitude to him for his endless efforts. I also want to thank Dr. Antony Franklin A for his support and guidance in my thesis work. I would like to thank all my peer members who indirectly or directly helped me towards building a research atmosphere at NeWS Lab. Foremost, I thank Anil Rangiseti, Amogh P.C, and Harsh Agarwal for helping me out in my thesis work. I consider having a fabulous time with my peer members of NMS research group for many technical concepts and social association. Mainly, I thank Dinesh Buswala, Mukesh Kumar Giluka, Gaurav Garg, Sumanta Patro, Debashisha Mishra, Venkatarami Reddy, and Veerendra Kumar Gautam for the consistent help and guidance throughout my journey. I'm also thankful to Mohd. Saalim Jamal and Himank Gupta for helping me in my tough times.

Dedication

To,
My Country,
My Parents and My Brother

Abstract

With the advances in cloud computing, the telecom operators are moving towards virtualization and cloud model. The future mobile networks are going to be deployed in the cloud with the help of Software Defined Networks (SDN) and Network Functions Virtualization (NFV) technologies. This thesis discusses various aspects related to prototyping and orchestration of slicing in LTE and 5G core networks using open-source technologies. The network slices are created in the user plane of the core network for differentiating the user traffic in terms of their QoS requirements.

The Service Based Architecture (SBA) of 5G core is realized with the help of Google Remote Procedure Call (gRPC) as Service Based Interface (SBI) and Consul as Network Function Repository Function. On top of the setup created, the concept of Look-aside Load Balancing is used, to suit the 5G SBA and efficiently handle the incoming traffic load. With the Access and Mobility Management Function as a use case for load balancing, it is concluded that carefully chosen load balancing algorithms can significantly lessen the control plane latency when compared to simple random or round-robin schemes.

Contents

Declaration	ii
Approval Sheet	iii
Acknowledgements	iv
Abstract	vi
Nomenclature	viii
1 Introduction	1
1.1 Cellular Networks	1
1.2 Cellular Network Architecture	1
1.2.1 LTE Architecture	1
1.3 5G Core Network Architecture	3
1.4 Key Enablers for future Mobile Networks	4
1.4.1 Software Defined Networking	5
1.4.2 Network Functions Virtualization	5
1.5 Network Slicing	7
1.6 Thesis Organization	8
2 Auto-Scaling of Dataplane VNFs in LTE	9
2.1 Related Work	9
2.2 System Model	10
2.3 Bit rate Aware Auto Scaling (BAAS)	11
2.4 Simulation Setup and Results	12
2.5 Summary	14
3 Scalability and SLA Management of Network Slices	15
3.1 Related Work	15
3.2 Scalable Network Slicing Architecture	16
3.2.1 Network Slicing Profiler (NSP)	16
3.2.2 Network Slice Scaling Function (NSSF)	17
3.3 Implementation Framework	18
3.3.1 Scalable Dataplane for Network Slices	18
3.3.2 QoS Provisioning	19
3.3.3 Auto Scaling of eMBB and mMTC	19
3.4 Evaluation	20

3.4.1	Scenario-1 : Evaluation of the Proposed Bandwidth Policy Control Mechanism	22
3.4.2	Scenario-2 : Evaluation of Bandwidth Isolation and Scaling	22
3.5	Summary	24
4	Prototyping and Load Balancing the 5G Network Architecture	25
4.1	Related Work and Motivation	25
4.2	Background	26
4.2.1	gRPC	26
4.2.2	Consul	27
4.3	Realization of 5G-SBI	27
4.3.1	Motivation for gRPC	27
4.4	Realization of 5G-SBA	28
4.4.1	gRPC based 5G Core Architecture	28
4.4.2	Service Registration and Discovery	29
4.4.3	Call Flow among the NFs	30
4.4.4	Evaluation of Testbed	32
4.5	Load-Balancing Architecture for GRPC based 5G Core	33
4.5.1	Look aside load balancing	34
4.5.2	Implementation Framework	34
4.6	Evaluation of AMF LALB	36
4.6.1	Reduction of CP_l with increasing instances	36
4.6.2	Variation of CP_l with various load balancing schemes	37
4.7	Summary	37
5	Conclusions and Future Work	38
	References	39

Chapter 1

Introduction

1.1 Cellular Networks

Telecommunications Industry has achieved rapid advancements since the last few decades. In the earlier years of the cellular networks, they were used only for voice calls and text messaging. As technology advanced, they have been playing a crucial role in providing Internet connectivity around the world.

The current fourth-generation of cellular networks high-speed wireless communication for mobile users and thus enhances the user experience. The services provided by mobile network operators have become even more accessible and cheap. As a result, the number of users is drastically increasing every year and thus leading to growth in global mobile traffic.

Every year several new gadgets are introduced in the market in various form factors and increased capabilities and intelligence. According to the Cisco VNI forecast [1], the global mobile devices and connections grew to 8.6 billion in 2017, up from 7.9 billion in 2016. Globally, mobile devices and connections expected to grow to 12.3 billion by 2022 at a CAGR of 7.5 percent.

1.2 Cellular Network Architecture

A cellular network is a radio network distributed over land through cells where each cell includes a fixed location transceiver known as base station. These cells together provide radio coverage over larger geographical areas. User equipment (UE), such as mobile phones, is therefore able to communicate even if the equipment is moving through cells during transmission.

A core network is a telecommunication network's core part, which offers numerous services to customers who are interconnected by the access network. Its key function is to direct telephone calls over the public-switched telephone network. In general, this term signifies the highly functional communication facilities that interconnect primary nodes. The core network delivers routes to exchange information among various sub-networks.

1.2.1 LTE Architecture

2G & 3G use packet switched core for data and traditional circuit-switched core for voice & SMS. That means the operator has to maintain two core networks simultaneously. A significant change

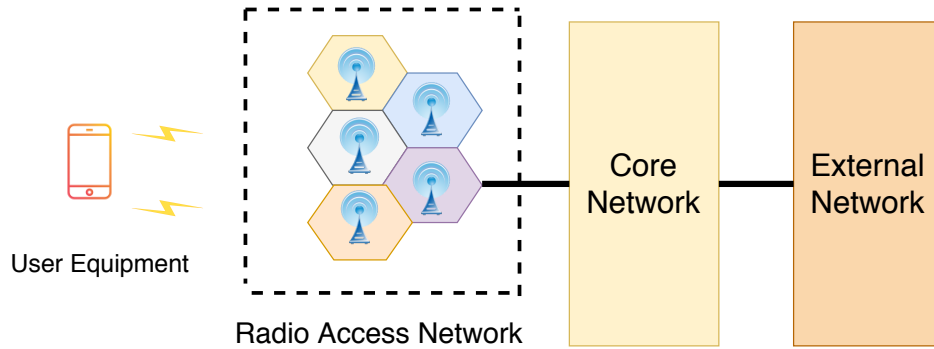


Figure 1.1: Cellular Network Architecture.

in LTE was the adoption of all IP based core network, which dramatically simplifies the design and implementation of the air interface. QoS mechanism on all interfaces ensures that Bandwidth and other requirements of voice calls can be met even when the capacity limits are reached.

The high-level network architecture of LTE(refer Fig 1.2) is comprised of the following three main components:

- The User Equipment (UE): The UE shown in Fig. 1.2 is the LTE handset.
- The Evolved UMTS Terrestrial Radio Access Network (E-UTRAN): The E-UTRAN handles the radio communications between the mobile and the evolved packet core and has one component, the evolved base stations, called eNodeB or eNB. Each eNB is a base station that controls the mobiles in one or more cells. The base station that is communicating with mobile is known as its serving eNB.
- The Evolved Packet Core (EPC): The 3G technology uses both circuit switched network and packet switched network but in 4G the requirement is only packet switching. This leads an evolved core network which is referred to as EPC. Various nodes in the architecture are shown in Fig. 1.2. In order to reduce the latency of handovers all the decisions to be made with respect to handovers are handled by single node of eNodeB in the E-UTRAN. The EPC is fully packet switched and consists of various entities like Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving Gateway (S-GW), Packet Data Network (PDN), PDN-Gateway (P-GW) and Policy Control and Charging Rules Function (PCRF). eNodeB is connected to the MME for signalling procedure and S-GW for the user traffic. MME handles all authentication procedures, location update and all the issues related to mobility. It maintains a Tracking Area Update (TAU) to track the user mobility. HSS is a permanent database for storing the network operators subscribers. The SGW acts as a router, it handles the uplink and downlink user traffic between eNB and PDN. The Packet Data Network (PDN) Gateway (P-GW) communicates with the outside world i.e., packet data networks (PDN), using SGi interface. Each packet data network is identified by an access point name (APN). The Policy Control and Charging Rules Function (PCRF) is responsible for policy control decision-making, as well as for controlling the flow-based charging functionalities in the Policy Control Enforcement Function (PCEF), which resides in the P-GW.

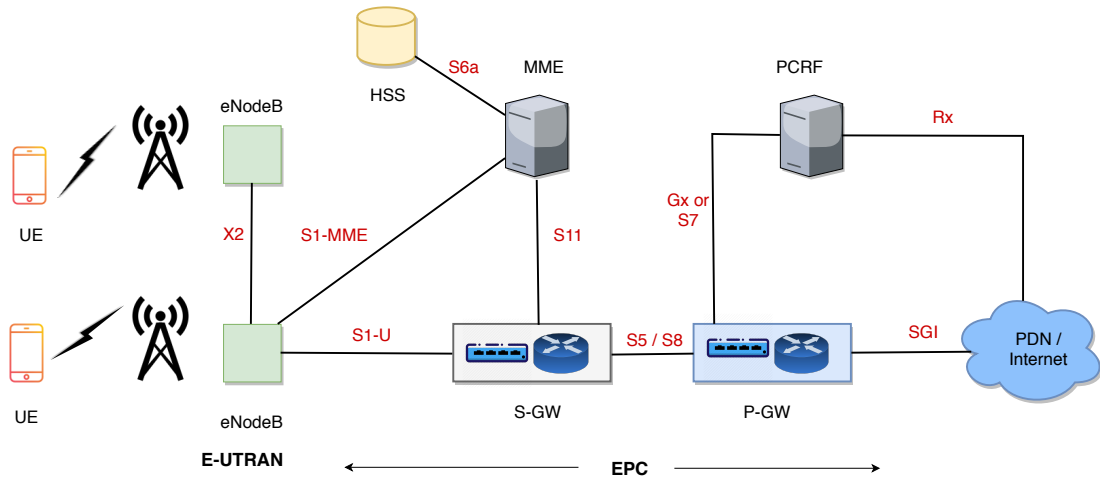


Figure 1.2: LTE Network Architecture.

1.3 5G Core Network Architecture

This section discusses the Service Based Architecture (SBA) proposed by the 3GPP in Release 15 [2] followed by brief descriptions of tools required to build it.

The 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.

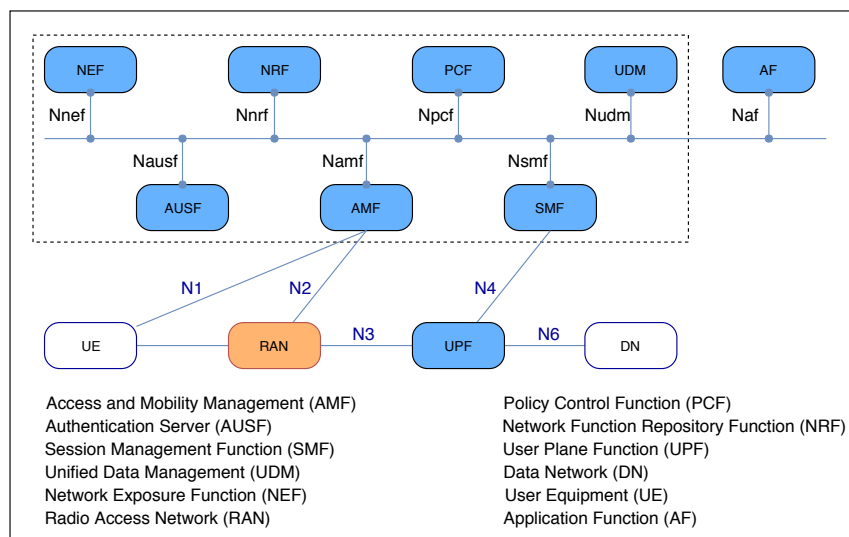


Figure 1.3: Service Based Architecture for 5G Core.

Fig. 1.3 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.

- Access and Mobility Management Function (AMF): AMF handles access control and mobility. If fixed access is involved, mobility management will not be needed in 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. 1.4. 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.

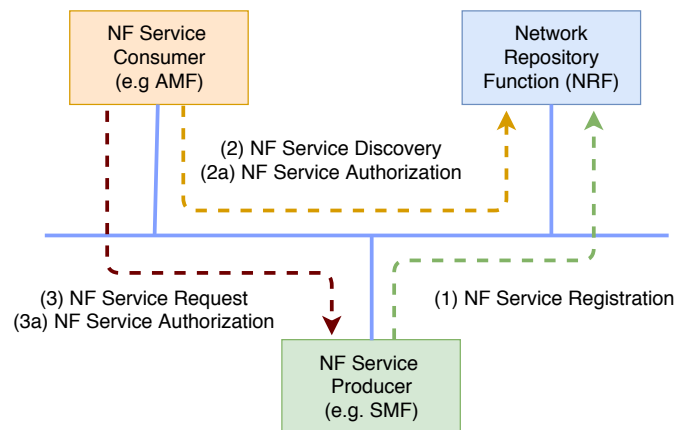


Figure 1.4: Service Registration and Discovery with NRF.

1.4 Key Enablers for future Mobile Networks

The digital transformation of network infrastructure through Network Functions Virtualization (NFV) and Software Defined Networking (SDN) is envisioned to play a vital role in the commercialization of 5G.

1.4.1 Software Defined Networking

SDN (Software Defined Network) has emerged as a networking model that has removed the limitations of current network infrastructures. The main goal of SDN is to make the network open and programmable. There is a logically centralized network Operating System that has a global view of all forwarding devices below it and has a way to communicate packet forwarding instructions to them called SDN Controller. The controller creates an abstraction or simplified view of the network applications. The Open Networking Foundation describes (ONF) a high-level architecture for SDN with three main layers, as shown in Fig. 1.5.

1. **Forwarding Devices / Data Plane:** These can be traditional hardware switches that support protocols like OpenFlow or software switches like OpenVSwitch. Hardware switches support higher performance, while software switches offer greater flexibility.
2. **South Bound Interface:** The SDN Controller needs a way to communicate with network forwarding devices. The South Bound Interface provides packet handling instructions, alerts of packet arrivals, status changes, statistics information, etc. to the SDN Controller using open flow protocol.
3. **SDN Controller / Control Plane:** Controllers typically run with key core services like Topology services, Inventory services, Statistics services, and Host tracking services. The topology services deal with how the forwarding devices are communicated together while the Inventory services keep track of basic information about SDN devices. Based on the traffic flowing in and out, the counters are updated, these services are Statistics services. The host tracking services track the IP address and MAC address of the host, to find where it is located.
4. **North Bound Interface:** The northbound application program interfaces (APIs) are the SDN REST APIs that interface the SDN Controller and the applications and services running over the network.
5. **SDN Applications:** The organization may have various network applications which can run on top of SDN like traffic engineering applications, security applications, Monitoring applications, Load Balancing, etc.

1.4.2 Network Functions Virtualization

NFV decouples the network services from the proprietary dedicated hardware and virtualizes the network services on commodity servers. 5G leverages technologies like SDN/NFV to efficiently handle the load at the 5GC.

ETSI NFV MANO Architecture

NFV Orchestration offers service coordination and instantiation, service chaining, scaling, service monitoring, etc. NFV MANO (Management and Orchestration) is an ETSI designed framework for management and orchestration of all resources in the cloud data centers. Fig. 1.6 shows the NFV architecture proposed by ETSI. It consists of the following components

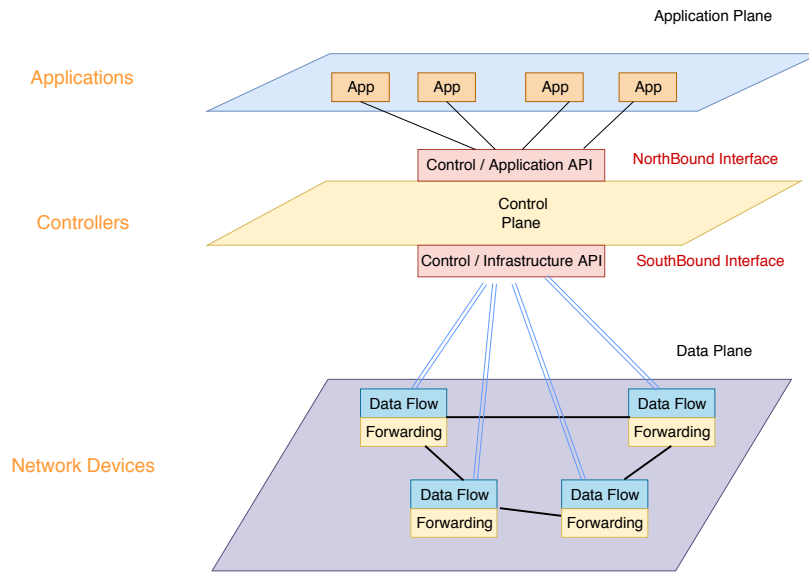


Figure 1.5: SDN Architecture.

- **Operations / Business Support Systems (OSS/BSS):** Software (occasionally hardware) applications that support back-office activities which operate a telcos network, provision and maintain customer services. OSS is traditionally used by network planners, service designers, operations, architects, support, and engineering teams in the service provider. Billing, order management, customer relationship management, and call center automation, are all BSS applications.
- **Element Management System (EMS):** The EMS in an NFV implementation provides network management of the Virtualized Network Functions (VNFs) and physical network elements (PNEs).
- **NFV Infrastructure (NFVI):** NFVI represents the underlying resources of the hardware like Compute, Storage, and Network. All these resources are virtualized and

The Management and Orchestration building blocks are

- **NFV Orchestrator (MANO):** NFVO is used for onboarding new Network Services and VNF packages. It takes care of Network Service lifecycle management which includes instantiation, scale-in/scale-out and termination of VNFs. It also does the global resource management, validation, and authorization of NFVI resource requests.
- **VNF Manager (VNFM):** VNFM looks after the lifecycle management of VNF instances. It is also responsible for adaptation role and overall coordination for configuration and event reporting between E/NMS and the NFVI.
- **Virtualised Infrastructure Manager (VIM):** VIM does the control and management of the NFVI compute, storage and network resources, within one operators infrastructure sub-domain.

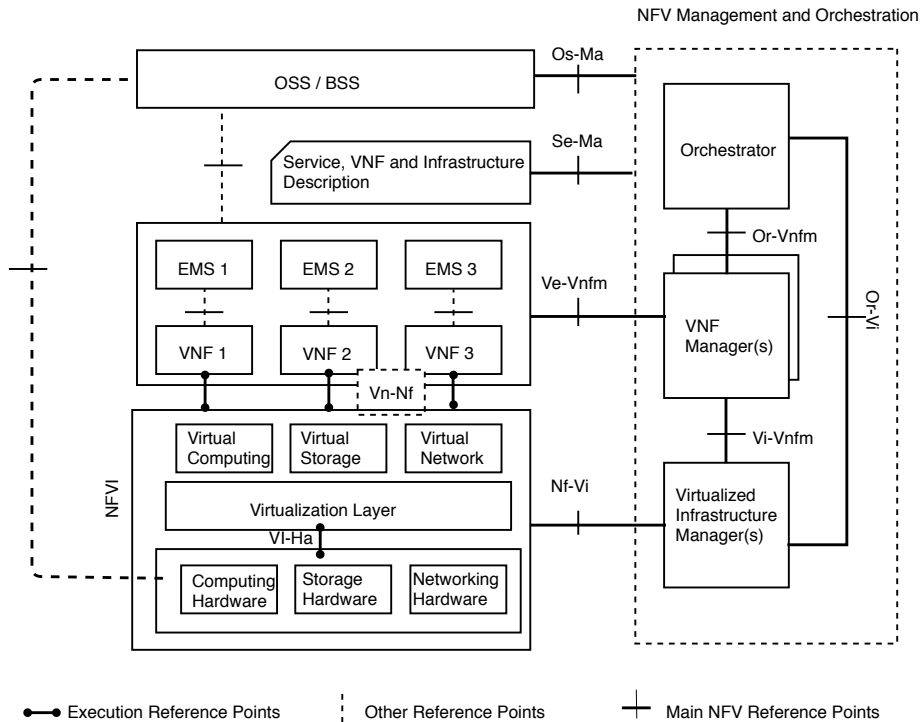


Figure 1.6: ETSI NFV Architecture.

1.5 Network Slicing

Since 5G serves a lot of use cases, it has diversified requirements for these use cases. Based on the requirements, the International Telecommunication Union (ITU), classified the use cases into three broad families, namely enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC), and ultra-Reliable Low-Latency Communications (uRLLC). The eMBB aims to focus on services that require high bandwidth and sustained high capacity network connections, such as virtual reality (VR), augmented reality (AR) and ultra high definition (UHD) videos. The uRLLC services are required for applications like Factory Automation, Tactile Internet, Intelligent Transportation Systems, Process Automation, etc., which have latency Constraints and require high reliability and availability. The mMTC enables the communications of the IoT devices over the cellular network.

While the human-generated traffic from mobile phone devices still exist, machine-type devices sending a small amount of information to other devices/clouds will account for a significant proportion and create massive MTC (mMTC) ecosystem. mMTC focuses on emerging smart services, including healthcare, manufacturing, utilities, consumer goods, smart agriculture, smart city, etc., which require a highly dense and large-scale deployment of MTC devices. All these use cases will share the same physical infrastructure at the mobile operators side. It would be costly to design a network with a single set of standard network functions that can handle a wide variety of use cases and at the same time, meet demanding performance requirements [1]. To support diversified use cases of 5G, there is a need for network architecture transformation, as the existing mobile network architecture was designed to meet requirements for voice and conventional mobile broad-

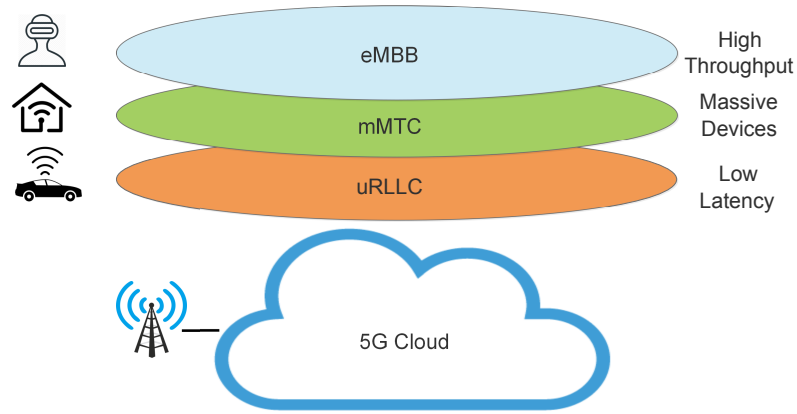


Figure 1.7: Network Slicing in 5G.

band (MBB) services. Network Slicing is the key to 5G network architecture evolution to support those above diversified 5G services. Network Slicing is a key concept that allows the mobile network operator (MNO) to split a single shared physical network into multiple logical or virtual networks. As these logical networks (Network Slices) are isolated, the failure of one slice does not affect the other slices. SDN & NFV capabilities, combined with cloud technologies, provide necessary tools to enable Network Slicing in 5G. Fig. 1.7 shows the requirements of various network slices.

1.6 Thesis Organization

The rest of the thesis is organized as follows.

Chapter-2 discusses the prototyping of network slices and scaling of data plane VNFs in LTE.

Chapter-3 explains the Service Level Agreement (SLA) management and scalability of the network slices using cloud platforms and open source orchestrators.

Chapter-4 discusses the realization of the Service-Based Architecture of 5G. It also explains the load balancing of network functions in the 5G Core.

Finally, **Chapter-5** concludes the work done and provides some future directions.

Chapter 2

Auto-Scaling of Dataplane VNFs in LTE

In this chapter, we present the realization of network slicing concept using LTE architecture. For network slicing based LTE architecture, Evolved Packet Core (EPC) needs to be designed with flexible and scalable NFV platforms with Virtualized Network Functions (VNFs) for Mobility Management Entity (MME), Serving Gateway (SGW), and Packet Data Network Gateway (PGW) running over a cloud platform to handle both control signal overhead and congestion in the data plane (DP). Besides, the orchestrator of NFV platform helps in monitoring the resource utilization, fault-tolerance, processing resources isolation, and security provisioning.

Our proposed work mainly deals with the creation of various core network slices using NFV based LTE architecture and show how to maintain specifically configured bit rates for each of the network slices using auto-scaling approaches. We also propose an auto-scaling approach for scaling of DP VNFs (S-GW/P-GW) and evaluate its performance in terms of network throughput and the number of active DPs needed for handling different traffic loads. However, scaling of resources of RAN and control plane of the core network is not in the scope of this work. Our main contributions in this work are:

- Implementation of a network slicing based LTE network architecture by extending NFV-LTE-EPC [3] framework.
- Proposed a Bit rate Aware Auto Scaling (BAAS) algorithm for auto-scaling of DP NFVs in the core network.
- Evaluated the network slicing based LTE architecture implementation and BAAS in terms of maintaining the bit rates of DPs and the number of DPs required.

2.1 Related Work

In [4], the authors discussed various coping technologies for efficient utilization of the concept of E2E network slicing in 5G mobile networks. The authors also proposed a Software Defined Mobile network Control (SDM-C) for E2E flexible network slicing. This work mainly describes a conceptual design

of the network slicing in 5G networks. In [5], the authors discussed various fundamental concepts like Network Functions (NFs), infrastructures, virtualization, orchestration, and isolation for designing network slices. Besides, the authors also summarized various challenges and benefits of network slicing realization using Open Networking Foundation (ONF) based architectures and NFV based architectures. Finally, SDN and NFV based network slicing architectures are proposed to overcome the challenges and provide maximum flexibility and scaling features to various E2E logical networks realized over 5G network architectures.

In [6], the authors proposed the creation of network slices based on QoS/security service requirements for various service level descriptions. Besides, they proposed a framework and mechanism to enable application services according to its high-level application service provider description. In detail, the framework defines concepts for application-specific slice selection and UEs associations and routing within the 5G network. However, the authors do not provide any system implementation details and evaluation results.

In [7], the authors developed and evaluated both SDN-based and NFV-based LTE-EPC implementations. In their study, various scenarios identified that an SDN-based LTE-EPC is well suited for handling a large amount of data traffic since it incurs minor overhead for forwarding packets from the kernel or switching hardware compared to an NFV setup where forwarding decisions are made in user space. On the other hand, an NFV based EPC system is well suitable for handling massive control signal overhead. In this work, we used the NFV-based LTE-EPC framework and extended it for network slicing to evaluate our proposed auto-scaling of DPs.

2.2 System Model

In the NFV-based LTE-EPC system, major control plane and data plane components such as MME, SGW, and PGW are implemented as VNFs. In our system model, scaling of control plane components and RAN are not discussed as it is outside the scope of this work. In network slicing system model, scaling of DP is defined as dynamically invoking SGW and PGW VNFs using lightweight virtualization platforms like Linux Containers (LXCs) [8] with the required network configuration of interfaces (S1-MME, S1-U, S5/S8 in core network).

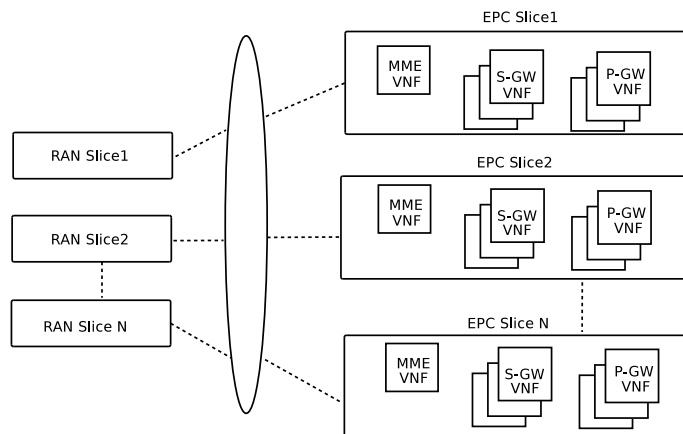


Figure 2.1: Network Slicing System Model.

As shown in Fig. 2.1, network slicing system (separate E2E network slices) is created over the NFV-based LTE-EPC architecture. Specifically, DP VNFs of the network slices are defined to meet specific Maximum DP Throughput (MDT) using processing resources. It means, the DP of a slice can provide only up to a throughput of MDT due to the limitation of the processing resources. To create a DP slice with a given MDT requirement, TCP Socket Buffer Size (TCP-SBSZ) implementation is used [9]. For a DP, TCP-SBSZ of SGW and PGW are set according to its required MDT. In specific, the TCP-SBSZ of SGW and PGW should be sufficiently large so that the flows through the slices can saturate the underlying DP path. When there is no competing traffic, the connections will be able to saturate the path if its TCP sender window is equal to the Bandwidth Delay Product (BDP) of the path. The BDP and its relation to TCP throughput and socket buffer sizing are well studied in the literature [9].

In our model, we use the MME to monitor the traffic loads of the slices and to allocate DPs with SGW and PGW. In each slice, the MME monitors the traffic load of DP and initiate auto-scaling of DPs. The MME module is extended with the proposed scaling approach to support auto-scaling of DPs.

2.3 Bit rate Aware Auto Scaling (BAAS)

We propose a Bit rate Aware Auto Scaling (BAAS) mechanism for auto-scaling of DPs. Timely monitoring of throughput of a DP is done by employing a network monitoring tool at the SGW VNF. For a network slice, the observed throughput of its DP is reported periodically by the SGW VNF to the MME which will then decide scaling-up (or scaling-down) of DPs for that particular network slice. Initially, a slice starts with a single DP to avoid over-provisioning. Then the MME scales-up the DPs of the slice when its throughput utilization (TU) reaches a threshold called TU_{THR} (*e.g.*, 90%). In our auto-scaling algorithm, TU is the scaling factor to scale-up DPs. Hence, UEs of the slice will not suffer from reduced bit rates. Setting higher TU_{THR} could lead to maximum utilization of the DPs. However, it could result in less available time to set up the DP. Choosing a lower value for TU_{THR} will lead to wastage of the DP resources. Hence, this value needs to be chosen carefully. We have opted for this as 90% in our experimental evaluation by keeping 10% as the buffer to contain fluctuations in the system load. So, BAAS approach tries to utilize each of DPs up to 90% of TU. For example, over the time, TU of some of the DPs might get lower than TU_{THR} , then BAAS instead of scaling new DPs, it selects a DP with maximum TU ($< TU_{THR}$) and uses it for serving any new UE arrivals.

Besides ensuring MDT requirement of a slice, another objective of BAAS is to minimize resource utilization of the slices without severely affecting the throughput of UEs. To reduce the overall resource consumption of the slices, auto scale-down of DPs is necessary whenever possible. As scaling down of active DP could result in an overhead of seamlessly transferring the flows of active UEs to other DPs, BAAS opportunistically wait for the completion of all UEs of a DP before shutting it down. Therefore, it does not cause any overhead in transferring flows of active UEs.

Algorithm 1 Bit rate Aware Auto Scaling (BAAS)

```
1: Runs at MME of the Network
2: Start each slice with single DP to avoid over provisioning
3: Monitor scaling factor TU of each DP in the slice
4: for each new UE request do
5:   TargetDP  $\leftarrow$  select a DP with maximum TU ( $TU < TU_{THR}$ ) in the slice
6:   Accommodate the UE in TargetDP
7:   if TU of TargetDP  $\geq TU_{THR}$  then
8:     Scale-up the DP by creating a new DP
9:     Deploy LXC of SGW and PGW for the new DP
10: Periodically runs auto scale-down:
11: ScaleDownDP  $\leftarrow$  select a DP with no active UEs in the slice
12: Shutdown the ScaleDownDP
```

2.4 Simulation Setup and Results

To implement the auto-scaling of DPs (SGW/PGW) in network slicing based 5G Architecture, we have extended the NFV based LTE-EPC implementation from [3]. In general, auto-scaling of either SGW or PGW is possible. But in our implementation, scaling of a DP results in scaling of both SGW and PGW, as the PGW in the NFV based LTE-EPC implementation is not handling any particular UE processing function. The NFV based LTE-EPC is extended to create network slices using VNF of MME, SGW, and PGW. The network slicing model used in our implementation is given in Fig. 2.1, which is realized using lightweight LXCs. Separate LXCs are used for deploying MME, SGW, and PGW. Each DP consist of an SGW and a PGW, and each DP of the slice is setup with a specific MDT. MDT of the DP is defined by setting up the TCP-SBSZ of read and write buffers of SGW and PGW. In our implementation, we created four network slices, Slice-1 to Slice-4 with MDT of 25 Mbps, 31 Mbps, 42 Mbps, and 52 Mbps, respectively. Fig. 2.2 shows various slices in the network and their MDTs.

The data traffic is generated using the tool “iperf3” that generates TCP traffic with a particular data rate and time duration. The traffic load can be varied by tuning the parameters of “iperf3”. In the simulations, DP throughput is measured from the “iperf3” output/statistics. We show that with different traffic load distribution, our proposed BAAS algorithm performs auto-scaling of DPs effectively in Slice-2. In our experimental scenario, there is enough bandwidth available for all DPs in the system. So the work did not include results capturing the influence of scaling-up (or scaling-down) of DPs in a slice on other slices in the network. Simulation parameters for traffic load distribution and Slice-2 specific parameters are given in Table. 2.1.

Fig. 2.3 shows the mean arrival rate (λ) of UEs in various time intervals. Fig. 2.4 describes the number of active UEs present in Slice-2 over the simulation time. Fig. 2.5 shows the observed network throughput over the simulation time. It is observed that during $t=0$ sec and $t=25$ secs, there is a sudden rise in the network throughput due to more UE arrival and it is causing TU of DP crosses TU_{THR} of 90%. Hence, it leads to scale-up of DP by adding a new DP (starting of a VNF of SGW/PGW) to maintain MDT of DPs. Now the number of DPs becomes 2, as observed from the Fig. 2.6.

After $t=100$ secs, Slice-2 is introduced with low traffic load by UE arrival rate with $\lambda=1$ and observed that the necessary scale-up and scale-down of DPs is done by BAAS (refer Fig. 2.6). As

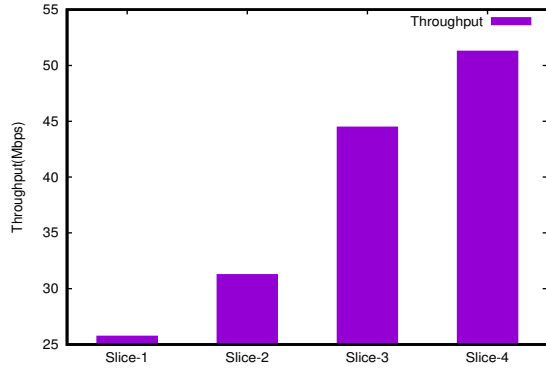


Figure 2.2: Maximum throughput observed in various slices.

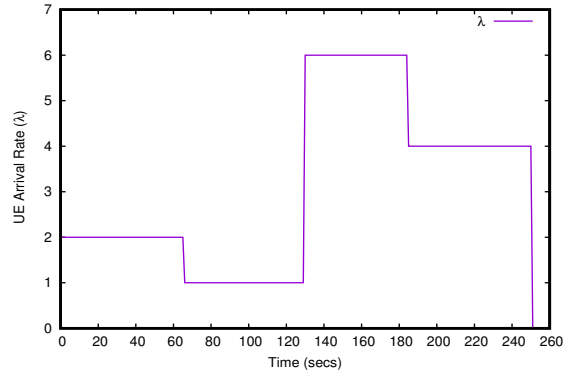


Figure 2.3: UE arrival rates (λ) at various time intervals.

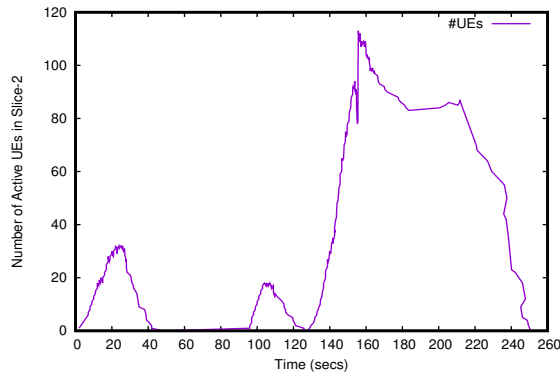


Figure 2.4: Number of active UEs in Slice-2 with time.

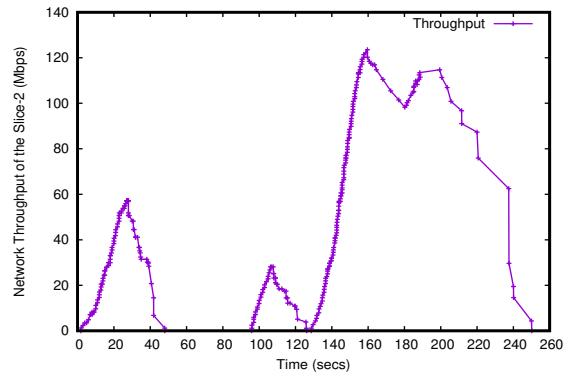


Figure 2.5: Network throughput of Slice-2 with time.

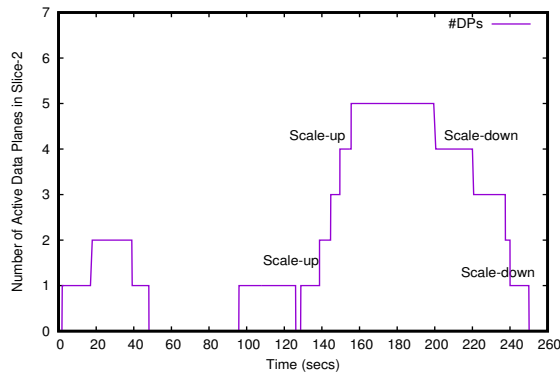


Figure 2.6: Number of active DP planes in Slice-2 with time.

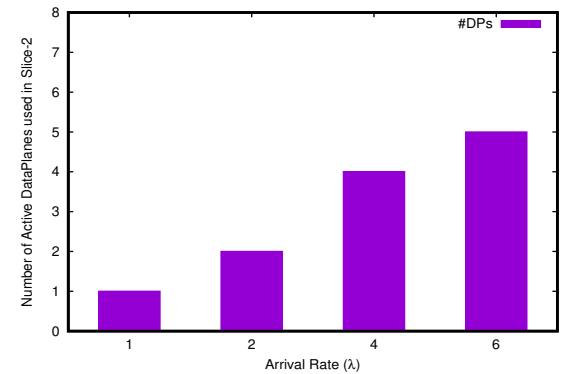


Figure 2.7: Number of active DP planes used in Slice-2 for different arrival rates.

shown in Fig. 2.4, after $t=130$ secs, the number of active UEs is increased and hence to meet this traffic load of UEs in Slice-2 and maintain MDT of each DP, BAAS started scaling up of DPs to

Table 2.1: Simulation Parameters

Parameter	Value
UE arrival	Poisson distribution
UE traffic duration	Uniform distribution (5-20 s)
MDT of DP in Slice-2	31.25 Mbps
UE arrival rate during 0-65 secs (λ_1)	2
UE arrival rate during 65-130 secs (λ_2)	1
UE arrival rate during 130-185 secs (λ_3)	6
UE arrival rate during 185-250 secs (λ_4)	4
UE traffic	iperf3 TCP
TCP-SBSZ of Slice-2 DP	128 KB
Scaling factor	TU
TU_{THR}	90%
Simulation time	260 secs

preserve individual MDT of DPs. In Fig. 2.6, between $t=130$ secs and $t=160$ secs, it can be observed that there are new active DPs scaled up and the number of DPs becomes 5. From $t=180$ secs, as the number of active UEs gradually decreased, DPs with no active UEs are getting shutdown by BAAS. As a result, we are effectively saving processing resource consumption due to unused DPs. In the Fig. 2.6, after $t=200$ secs, it can be observed that there are gradual scaling down of DPs. Finally, we summarize the number of active DPs in the network with the given arrival pattern of UEs in Fig. 2.7. It is clear that BAAS dynamically vary the number of active DPs based on the load in the network. Without the BAAS mechanism the network would either suffer from reduced bit rates or over-provisioning of the underlying resource.

2.5 Summary

In this work, we demonstrated the network slicing using LXC. We proposed an auto-scaling approach called Bit rate Aware Auto Scaling (BAAS) for maintaining bit rates of DPs and avoiding over-provisioning of underlying DP VNFs for network slices. Unlike general auto-scaling approaches, which are based on processing resources utilization, BAAS builds on the actual throughput utilization of DPs. Hence, it tries to scale-up new VNFs of DPs to maintain MDT.

Chapter 3

Scalability and SLA Management of Network Slices

In the previous chapter, we deployed the Dataplane Network Slicing using the SGW and PGW components on the lightweight Linux containers and did the auto-scaling of data plane using the MME Virtual Network Function (VNF). The auto-scaling is done based on the network throughput requirements of each network slice. In this chapter, we mainly focus on the operational aspects of the network slicing.

One of the significant challenges in the realization of Network Slicing is going from a high-level description of network slice to concrete slice implementation in terms of infrastructure and network functions. There is a need for comprehensive orchestration of different network slices in a way that each slice meets its SLA requirements and also efficiently utilizes underlying resources. In this chapter, we present a Scalable Network Slicing Architecture (SNSA) to tackle the above challenges.

3.1 Related Work

In [7], the authors developed an NFV-based LTE EPC [3] implementation. In this implementation, core network functions of LTE are deployed in virtual machines running in a private cloud environment. They simulated the working of a typical EPC for handling signaling and data traffic across the multiple virtual machines. In [10], the authors presented a design of a flexible 5G architecture with the emphasis on techniques that provide efficient utilization of substrate resources for network slicing. The paper proposes the network slicing aware orchestration framework to perform service and resource orchestration by considering various slice requirements. It also considers the QoS aware Network Orchestration and Control to meet the Slice Configuration. In [11] authors proposed a vision for LightEPC to orchestrate on-demand creation of light weight mobile packet cores for Machine Type Communication (MTC) traffic and simplify network attach procedure of MTC devices. By doing so, LightEPC can create and scale instances of NFV MTC functions on demand and in an elastic manner with any sudden increase in MTC traffic.

Our main contributions in this chapter are summarized as follows:

- Scalable Network Slicing Architecture based on nonstandalone 5G architecture

- Implementation of Network Slicing Profiler and Network Slice Scaling Function for flexible orchestration of resources.
- Evaluation of the performance of various network slices in terms of bandwidth isolation, scalability, and SLA management.

3.2 Scalable Network Slicing Architecture

In [12], Next Generation Mobile Networks (NGMN) Alliance proposed a layered representation for network slicing, where the three layers are Service Instance layer, Network Slice Instance layer, and Resource layer. The Service Instance layer represents the end user services supported by a slice. The instances of the services will reside in the Network Slice Instance layer, and Resource layer represents all the resources from the underlying network infrastructure. In [13], the authors discussed the Network Slicing in the NFV framework. They did a mapping of the NGMN layering concept to ETSI NFV architectural framework. In this work, we present Scalable Network Slicing Architecture (SNSA), an NFV based network slicing concept by proposing a novel Network Slicing Profiler (NSP) and Network Slice Scaling Function (NSSF) as additional components to the above architecture. Fig. 3.1 shows our proposed SNSA, which includes NSP and NSSF.

3.2.1 Network Slicing Profiler (NSP)

The Adaptive Management and Orchestration (MANO) is crucial in ensuring the performance requirements of the deployed services [14]. It should be efficient at utilizing underlying resources by making decisions based on the current state of slices as well as their predicted demands shortly. As the network slices share the same underlying NFV Infrastructure (NFVI), there is a need to design adequate resource management mechanisms, that maintain isolation among slices and also meet the performance requirements of the slices [5].

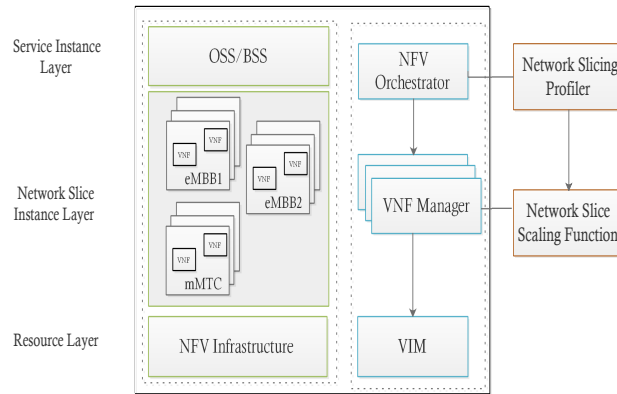


Figure 3.1: Scalable Network Slicing Architecture

To address the above challenges, we propose a novel NSP in our architecture. The NSP maintains profiles of various network slices concerning set of available physical and virtual network resources. These profiles help in an efficient allocation of resources to network slices. A new profile can be created based on the SLA requirements of the network slice that needs to be deployed. The requested resources are allocated by the NSP to the network slice based on its profile. The NSP keeps track

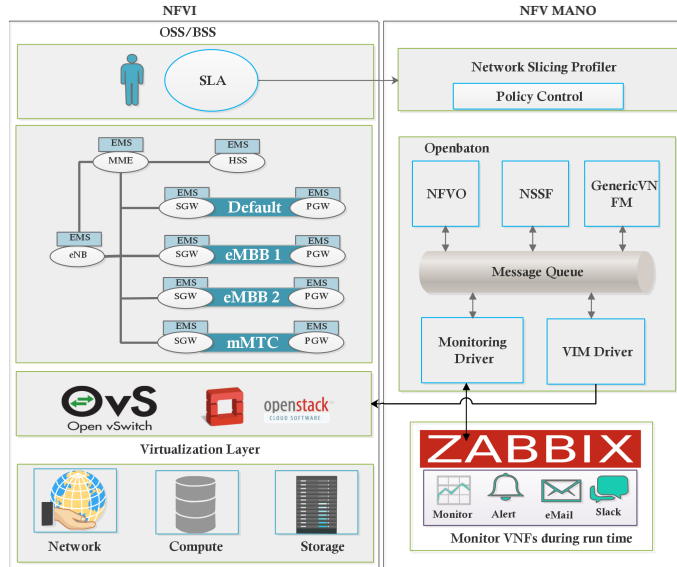


Figure 3.2: Network Slicing Implementation Framework

of the resource usage by various network slices. Depending on the workload, the NSP dynamically reallocates the resources to the network slice. The concept of dynamically increasing resources to a given slice on runtime is called Vertical Scaling [15]. The Vertical Scaling should have minimal impact on the services of this slice and the other slices in the network. The operators need to define the maximum amount of resources that can be allocated to each slice in the NFVI components. The NSP limits the resources that are being allocated in a single Host server in the NFVI. In this way, the isolation is maintained by the NSP in terms of resource allocation. If the network slice requires resources more than a maximum limit of the resources, then NSP triggers the NSSF.

3.2.2 Network Slice Scaling Function (NSSF)

Scaling of the resources is required to cope with the resources required for peak traffic. The scaling of the network slice is one of the crucial capabilities when the network slice is in production. The NSSF involves the creation of new VNFs of the slices by monitoring various metrics of the VNF. The process of monitoring data and triggering of scaling events must be programmable and automated. The triggering for scale-up may happen upon various conditions depending on the type of the network slice. For example, if a slice is of type eMBB, to ensure minimum bandwidth to the UEs, it has to scale based on the bandwidth consumption of the slice. As the resource allocation by NSSF requires the creation of new VNFs, it incurs an overhead of booting and setup. So appropriate thresholds have to be set to keep the service continuity. Scaling in the network slice can occur at different levels:

- Scaling of particular VNF
- Scaling of particular network service (collection of VNFs)

NSSF maintains the resource pool of various hosts and allocates them on demand. In this way, the NSP and NSSF help MANO Systems in utilizing the resources efficiently and also ensuring isolation among the slices.

3.3 Implementation Framework

In this section, SNSA is realized with the help of open source tools. The use of open-source technology provides a wide range of open development models to large operators and enterprises. Various platforms that are used to realize SNSA is shown in Fig. 3.2. OpenStack [16] is an open-source cloud virtualization platform. OpenStack project offers the ability to design the NFV system to build large and scalable services. Open Baton [17] is an open-source implementation based on ETSI NFV MANO architecture implemented in Java J2EE framework. It provides modular and extensible architecture based on the Advanced Message Queuing Protocol (AMQP) [18] protocol realized by RabbitMQ [19]. The significant components of Open Baton are NFV Orchestrator (NFVO), Generic VNF manager (VNFM), and Auto Scaling Engine (ASE). Communication among these various components happens via RabbitMQ. Open Baton provides the plugin mechanism to communicate with cloud environments like Docker [20], OpenStack, etc. In this work, the OpenStack plugin mechanism is used. Open Baton integrates with a monitoring system via the monitoring plugin. In this work, the default Zabbix plugin is used, as Zabbix monitoring system [21] is being implemented for VNF monitoring. A Network Service Descriptor (NSD) file is created in JSON format which contains the network slicing setup. Fig. 3.3 shows the interaction among the various components of our setup. Open Baton NFVO uses the NSD to launch the setup. This is done either via the dashboard in the web browser or via command line interface. Once the NSD is launched, the VNFs are created, and links are setup among the VNFs on top of OpenStack. Open Baton NFVO interacts with the Generic VNFM to create new VNFs. The Element Management System (EMS) is responsible for the performance and fault management of VNF.

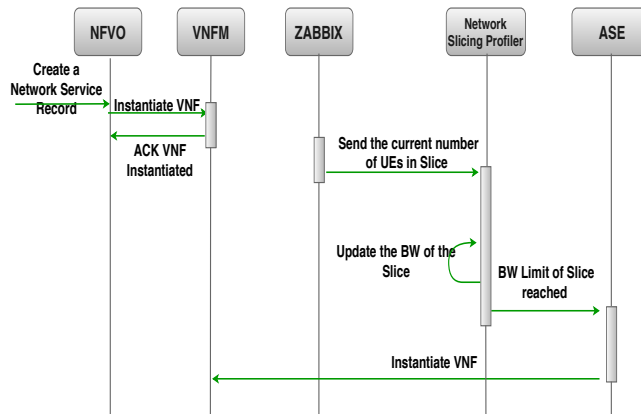


Figure 3.3: Workflow between various components.

3.3.1 Scalable Dataplane for Network Slices

The NFV-LTE-EPC [3] is extended to build SNSA. The NFV-LTE-EPC implements the standard 3GPP protocol stacks for EPC modules composed of MME, HSS, SGW, PGW including a RAN simulator and a Sink node. The purpose of RAN simulator is to generate uplink and downlink data to EPC. The RAN simulator produces multiple threads and creates a RAN object for each slice of eMBB and mMTC, which in turn leads to control plane and data plane transmissions with EPC. The Sink node module is used to serve as a PDN server to receive the generated uplink

traffic and send back acknowledgments as downlink traffic. The MME uses the Stream Control Transmission Protocol (SCTP) for its connection with eNodeB and HSS. The core network consists of four data plane network slices: one default slice, two eMBB slices, and an mMTC slice. A data plane slice consists of the logical network by arranging SGW and PGW together. A separate sliceID is allocated for each slice. During the UE registration procedure, it connects to the default slice for the authentication and assigns an IP address to UE. The default slice validates whether the UE is authorized to the 5G core network. The default slice selects network sliceIDs for corresponding network sliceIDs which are obtained from the UE. The UE sends user data based on allocated IP address and network sliceID and sends data through the data traffic path.

3.3.2 QoS Provisioning

QoS conditions agreed under SLAs have to be fulfilled by the network slices of eMBB, uRLLC, and mMTC. 5G Dataplane of eMBB requires higher per UE throughput, and bandwidth of the eMBB should be allocated on demand as bandwidth is expensive. mMTC slices are different from eMBB and uRLLC slices as it does not possess strict requirements on the bandwidth and latency. The IoT devices exhibit different traffic patterns such as Periodic, continuous, delay based, etc. mMTC slices should be able to scale data pattern, signaling pattern, burst rate, activity cycle, etc. The resources for mMTC slice have to be allocated based on the number of devices and their requirements. The slices have to expand and contract based on their needs.

Open Baton provides NFV complaint external component called Network Slicing Engine (NSE). It is realized using spring.io Java Framework. It communicates with NFVO via Open Baton's SDK. The QoS parameter is defined in NSD for the network slices. The default NSE in Open Baton allows only three types of parameters GOLD (200-150 Mbit/s), SILVER (100-50 Mbit/s), and BRONZE (50-25 Mbit/s) for Bandwidth. It cannot dynamically allocate QoS based on the requirement of the slice. As we have to set the QoS based on the slice type dynamically, we created NSP. The NSP is implemented in Python. It is integrated with OpenStack, Open vSwitch (OvS) [22], and Open Baton. OvS is an open source software implementation of a virtual multilayer switch. It has been integrated with as a software-based control stack for dedicated switching. To enable QoS service in OpenStack, the steps in [23] are followed. The QoS policy is created, and the bandwidth limit rule is set as per the policy using NSP.

3.3.3 Auto Scaling of eMBB and mMTC

Our proposed NSSF is realized with the help of the Open Baton's ASE. ASE does auto-scaling of single VNF by considering the load of the VNF. Here in NSSF, the ASE is extended to perform the scaling of the set of VNFs. The auto-scaling of VNFs can be performed based on various metrics obtained by continuous monitoring of VNFs by Zabbix. The various metrics can be load on CPU, percentage of CPU utilization, incoming/outgoing traffic at the network interface, etc. As there are two types of slices (eMBB & mMTC) in our system, for an eMBB type of slice, NSP triggers the ASE to create a new instance when the maximum bandwidth limit of the slice is reached. For mMTC type of slice, an auto-scaling metric of CPU load on the VNF averaged over one minute is considered. The CPU load is a measure of the amount of computational work that a VNF performs. This is typically measured over a period of one, five, and fifteen minutes. Auto-scaling in mMTC

is done by considering the CPU load of VNF averaged over one minute time duration with the cool down period of ten seconds to avoid rapid scale up and scale down of the slices.

Table 3.1: VNF Specifications

Entities	VM Flavor	#vCPUs	RAM
RAN, SINK	m1.medium	2	4GB
MME, HSS	m1.small	1	2GB
SGW, PGW(default slice)	m1.large	4	4GB
SGW, PGW(eMBB slice)	m1.large	4	4GB
SGW, PGW(mMTC slice)	m1.small	1	2GB

3.4 Evaluation

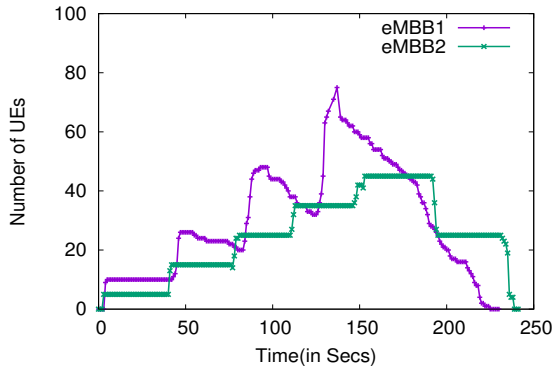


Figure 3.4: Number of UEs with Time.

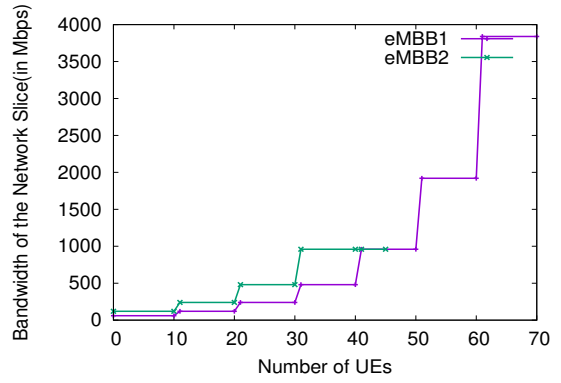


Figure 3.5: Bandwidth Provisioned for each Slice.

The experimental setup is developed by extending NFV-LTE-EPC [3]. Each of the core network components HSS, MME, SGW, PGW are implemented as multithreaded server modules. The RAN simulator in NFV-LTE-EPC [3] has the functionality of UE and eNodeB for generating the traffic to the core network. The Open Baton as the NFVO and OpenStack as a virtualized platform with OvS are used to build ETSI aligned SDN NFV platform. Our testbed setup runs on a private OpenStack

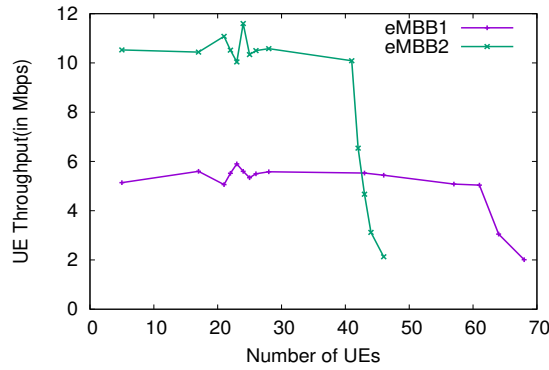


Figure 3.6: Individual UE throughputs in eMBB slice.

cloud that can be controlled and managed via OpenStack API or Horizon dashboard. The single node OpenStack machine running Newton release is used for our test scenarios. Zabbix Agent in each VNF is configured for real-time monitoring and tracking the consumed network resources of instances deployed on OpenStack.

To evaluate our test scenarios, the simulation parameters given in Table 3.2 are considered. The RAN simulator of NFV-LTE-EPC [3] is used to generate continuous control and data plane traffic to the EPC. We simulate concurrent UE threads of eMBB and mMTC on the RAN simulator. The performance of test scenarios is measured using the following key performance metrics:

1. Average UE Throughput: average UE Throughputs obtained from iperf.
2. Bandwidth Provisioned: the amount of bandwidth that is allocated to a slice.
3. Number of slice instances: The total number of slice instances created with auto-scaling.
4. CPU load: It is CPU load of the VNF averaged over one minute.

Table 3.2: Simulation Parameters

Parameter	Value
Number of UEs	0 to 300
Simulation time	360 seconds
Network Slices	[eMBB1, eMBB2, mMTC]
SliceIDs	[s1,s2,s3]
Packet Size	[800, 800, 100] Bytes
Min. Bandwidth for flow	[5, 10, -] Mbps
UEs_{init}	[10,10,-]
BW_{init}	[60 Mbps, 120 Mbps, 500 Mbps]
BW_{max}	[1920 Mbps, 960 Mbps, 500 Mbps]
UE Data Transfer Duration	[60-180s, 30-80s, 5s]
UE Arrival Distribution	Poisson Distribution
Mean Arrival Rate(λ) for eMBB1 [0:250s]	12
Mean Arrival Rate(λ) for eMBB2 [0:150s]	12
Mean Arrival Rate(λ) for mMTC [0:50s]	4
Mean Arrival Rate(λ) for mMTC [100:150s]	15
Mean Arrival Rate(λ) for mMTC [150:200s]	8
Virtualization platform	OpenStack Newton
NFV orchestrator	Open Baton 5.1.2
Live status monitor	Zabbix 3.0.0

To evaluate the effectiveness of the proposed network slicing architecture, two different scenarios are presented.

- Scenario-1: Evaluation of the Proposed Bandwidth Policy Control Mechanism
- Scenario-2: Evaluation of Bandwidth Isolation and Scaling

3.4.1 Scenario-1 : Evaluation of the Proposed Bandwidth Policy Control Mechanism

Algorithm 2 Network Slicing Profiler for eMBB

```

1: procedure BANDWIDTH_PROVISIONING( $UES_{init}, BW_{init}$ )
2:   while True do
3:      $UES_{in\_slice} = \text{currentNumberOfUEs}()$ 
4:     if  $UES_{in\_slice} \% UES_{init} = 0$  then
5:        $n = UES_{in\_slice} / UES_{init}$ 
6:       Update Slice\_BW\_Policy =  $2\hat{n}-1 * BW_{init}$ 

```

This scenario evaluates our approach of automated bandwidth policy control shown in Algorithm 2. The Algorithm 2 starts with an initial number of UEs (UES_{init}) which has initial bandwidth of BW_{init} and the algorithm doubles the current bandwidth of the slice in every UES_{init} . The current number of UEs in the slice is obtained by the Zabbix monitoring of the VNFs of the slice. The values of UES_{init} and BW_{init} depend on the SLA requirements of the eMBB slice. The UES_{init} and BW_{init} values of eMBB1 and eMBB2 are given in Table 3.2. Fig. 3.4 shows the increase of UEs in the slices with time. Fig. 3.5 shows the dynamic bandwidth provisioning of Algorithm 2 with the increasing number of UEs. As a result of automated bandwidth provisioning, Fig. 4.10 shows the individual UE throughputs concerning an increasing number of UEs. These UE throughputs lie in the range of 5 Mbps to 6 Mbps up to 60 UEs (for eMBB Slice-1) and in range of 10 Mbps to 11.5 Mbps up to 40 UEs (for eMBB Slice-2). Then after that, we observe a drop in the per UE throughput, as we limit the maximum bandwidth that can be provisioned for a slice. Beyond this limit, the bandwidth cannot be guaranteed to the flows. So, there is a need to instantiate new VNFs and new links to handle the number of UEs beyond the limit. Hence, there is a need for scaling the VNFs after reaching the Bandwidth limit of the Slice. The following section shows the scaling and maintaining the isolation among the slices.

3.4.2 Scenario-2 : Evaluation of Bandwidth Isolation and Scaling

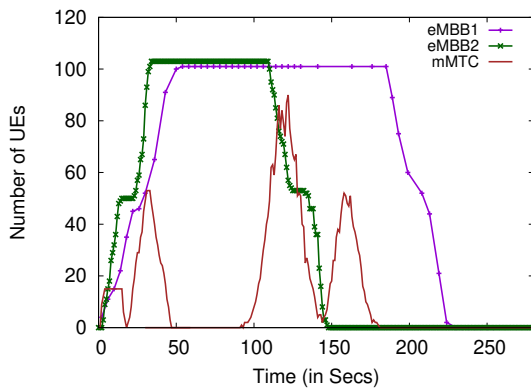


Figure 3.7: UE Load distribution over simulation Time.

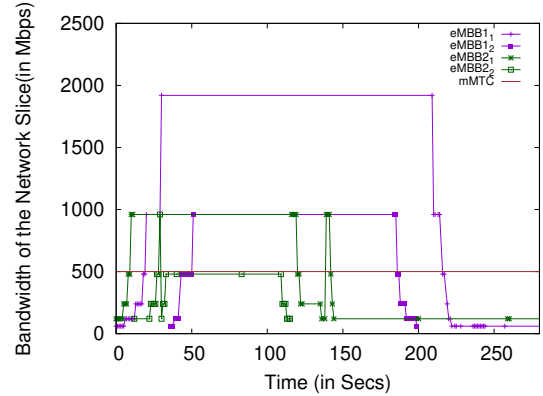


Figure 3.8: Bandwidth Provisioned for Slice over Time.

This experiment is to show how our testbed guarantees bandwidth isolation and auto-scaling using NSSF for the network slices. NSSF uses the Algorithm 3 to scale the eMBB and mMTC slices.

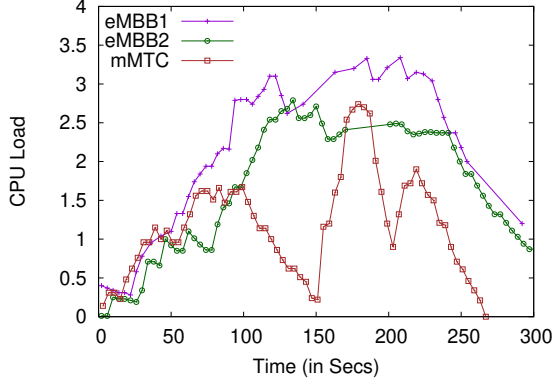


Figure 3.9: CPU Load Average over all the Instances with Time.

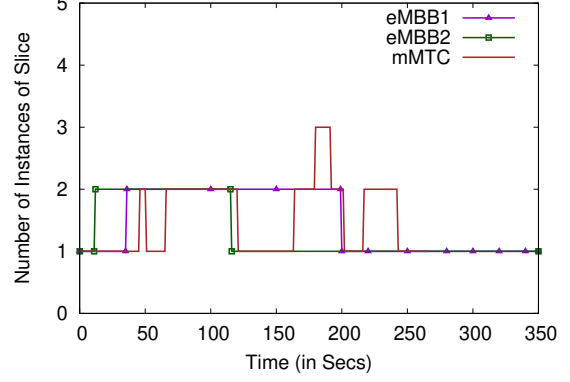


Figure 3.10: Number of Slice Instances (SGW+PGW) over Time.

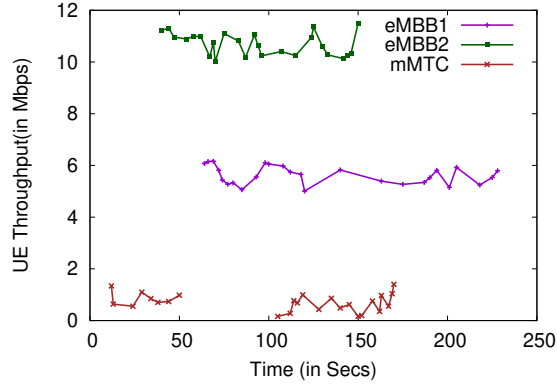


Figure 3.11: Average Per UE Throughput observed over Time.

The Algorithm 3 takes parameters of sliceID and sliceType. SliceID uniquely identifies various network slices in the system. The sliceIDs of eMBB1, eMBB2, and mMTC are given in Table 3.2. SliceType indicates whether the slice is of type 'eMBB' or 'mMTC'. The eMBB type slice is scaled based on the maximum bandwidth (BW_{max}) of the slice. The BW_{max} values of various slices are mentioned in Table 3.2. Since mMTC type slices need not provide any minimum bandwidth guarantees to UEs, it need not scale based on the bandwidth of the slice. The mMTC type slices are scaled based on the CPU load of the VNFs in the slice. The CPU load of the VNFs will be same in the slice, as all the VNFs in the slice are only doing data forwarding. The current bandwidth consumed by a slice and CPU load of the VNFs is fetched by Zabbix monitoring. The ScaleUp and ScaleDown operations are performed by the Open Baton's ASE. We simulate concurrent UE threads of eMBB1, eMBB2, and mMTC using RAN-Simulator with the traffic load as shown in Fig. 3.7. Poisson distribution is used for modeling the UE arrival rate, with the mean arrival rates mentioned in Table 3.2. To meet the minimum bandwidth guarantees to the UEs of eMBB slices, NSP uses Algorithm 2 and provisions the bandwidth, as shown in Fig. 3.8. The NSSF scales up to the eMBB1 slice as shown in Fig. 3.10 during time $t=36$ sec, to guarantee minimum per UE throughput for an eMBB1 slice. Similarly, NSSF scales up eMBB2 slice during time $t=12$ sec. As load decreases,

Algorithm 3 Network Slice Scaling Function

```
1: procedure AUTO_SCALING(sliceID, sliceType)
2:   if sliceType = “eMBB” then
3:      $BW_{current} = \text{currentBandwidth}(\text{sliceID})$ 
4:     while True do
5:       if  $BW_{current} = BW_{max}$  then
6:         ScaleUp(sliceID)
7:       if  $BW_{current}$  less than  $BW_{init}$  then
8:         ScaleDown(sliceID)
9:   if sliceType = “mMTC” then
10:     $cpuload_{current} = \text{currentCPULoad}(\text{sliceID})$ 
11:    while True do
12:      if  $cpuload_{current} = 1.0$  for ten sec.s then
13:        ScaleUp(sliceID)
14:      if  $cpuload_{current}$  less than 0.1 then
15:        ScaleDown(sliceID)
```

NSSF scales down an eMBB1 slice at time $t=200$ sec and eMBB2 slice at $t=116$. As a result of network bandwidth profiling in Fig. 3.8 and auto-scaling in Fig. 3.10, it is observed that bandwidth of slices is unaffected and isolation of bandwidth is maintained. Also Fig. 3.11 shows an average per UE throughput of 5 Mbps for eMBB1 and 10 Mbps for eMBB2 slices that meet the minimum requirements of UEs in eMBB1 and eMBB2, respectively.

Fig. 3.9 shows the total CPU load of three slices over all the instances. We observe that as the number of UEs increase, the CPU load is also increasing beyond the capability that a VNF can handle. So, for mMTC type slices, the auto-scaling metric is chosen as the CPU load. We set the auto-scaling threshold of CPU load as 1.0 as it is an ideal threshold. Continuous monitoring of the CPU load of mMTC slice is done and once the CPU threshold is reached, NSSF waits for cooldown period of 10 secs for checking if the CPU threshold is consistently above 1.0. If it is, then Open Baton’s ASE scales up the new instance, as shown in Fig. 3.10.

3.5 Summary

The next generation mobile network will need network slicing to meet various use cases. In this work, we designed and developed a novel Network Slicing Profiler, and Network Slice Scaling Function used the open source technologies to realize the network slicing environment. The bandwidth isolation among slices and scaling of the slices are evaluated by considering the three slices, two of type ‘eMBB’ and one of type ‘mMTC’. It is also demonstrated that SLAs of the eMBB slices are ensured when all three slices are running. To the best of our knowledge, this is the first attempt to show the scalability of various network slices in an orchestrated environment.

Chapter 4

Prototyping and Load Balancing the 5G Network Architecture

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) [24], 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[25], an open-source distributed and highly available service discovery and configuration system.

We propose using a look-aside load balancer [26] 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 a 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 UE traffic load with different load balancing strategies.

4.1 Related Work and Motivation

The authors in [27] 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 [28] 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 [29] 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 [30], 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 [7], 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 [31], 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 [32], 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, and 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. The main contributions of 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.

4.2 Background

This section discusses the tools required to build the Service Based Architecture (SBA).

4.2.1 gRPC

gRPC[24] is a modern, open source remote procedure call (RPC) framework widely used in microservices 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 [33] 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).

4.2.2 Consul

Consul [25] 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.

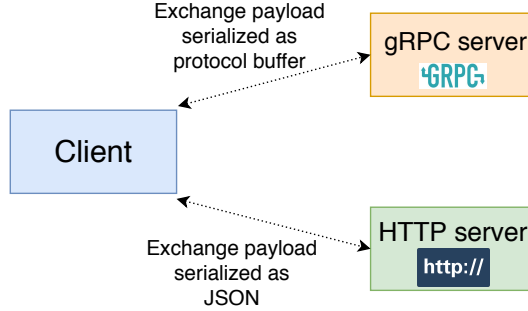


Figure 4.1: Benchmarking setup of gRPC and HTTP REST.

4.3 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 [28] have proposed and evaluated various protocol stacks that are shown in Table 4.1. In [28], the authors demonstrated HTTP 2.0 performs better than HTTP 1.1, and ProtoBuffers [33] 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 4.1: 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

4.3.1 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. 4.1. A gRPC

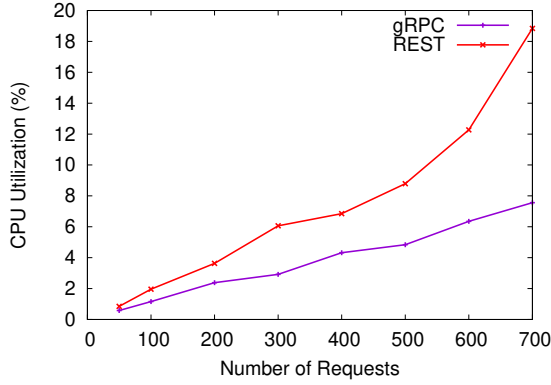


Figure 4.2: Comparison of CPU utilization of gRPC vs REST.

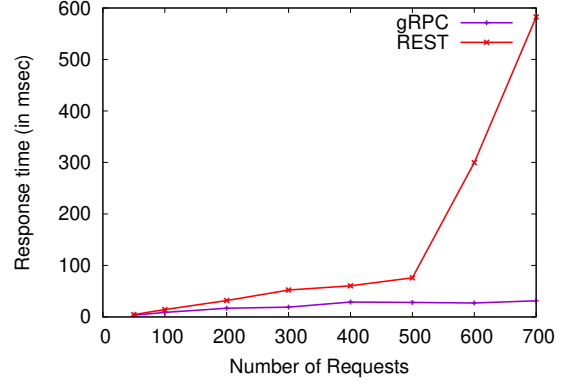


Figure 4.3: Comparison of Avg. response time of gRPC vs REST.

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. 4.2, we have plotted average CPU load on the gRPC and HTTP servers while increasing the number of clients. In Fig. 4.3, 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.

4.4 Realization of 5G-SBA

In this section, we describe the realization of gRPC based SBI for 5GC, followed by implementation of the NRF.

4.4.1 gRPC based 5G Core Architecture

Fig. 4.4 shows how our gRPC based 5G Core architecture (gRPC-5GC) fits in ETSI NFV architecture [34]. The virtualization of resources in the architecture is done with the help of Docker platform [20]. 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

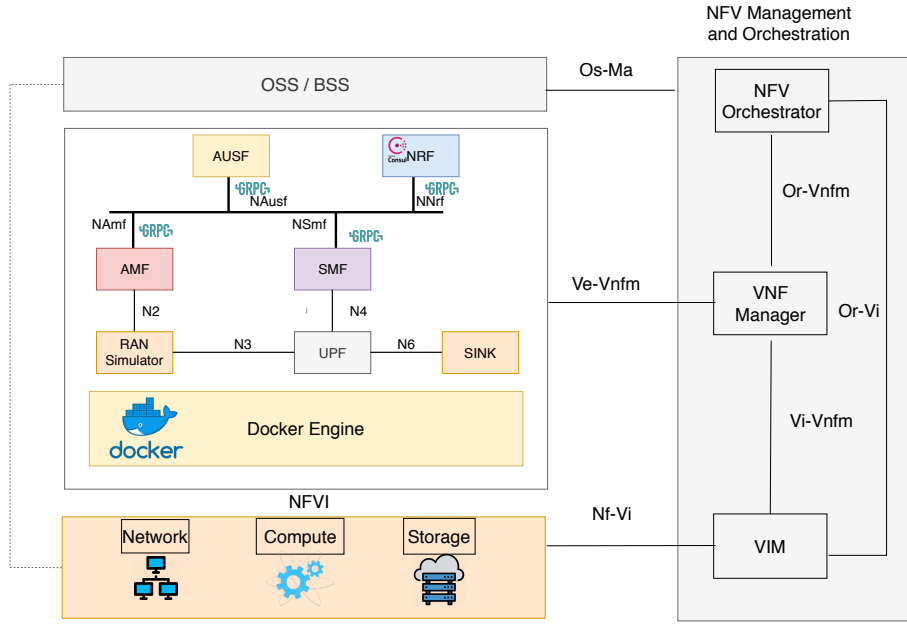


Figure 4.4: gRPC based 5G Core in ETSI NFV Framework.

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 acknowledgments 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. 4.4 are the reference point based links. Service-based ones follow the multi-threaded gRPC architecture. NAMf, NSmf, NAusf, and NNrf in Fig. 4.4 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 [35]. 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.

4.4.2 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. 4.5. 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

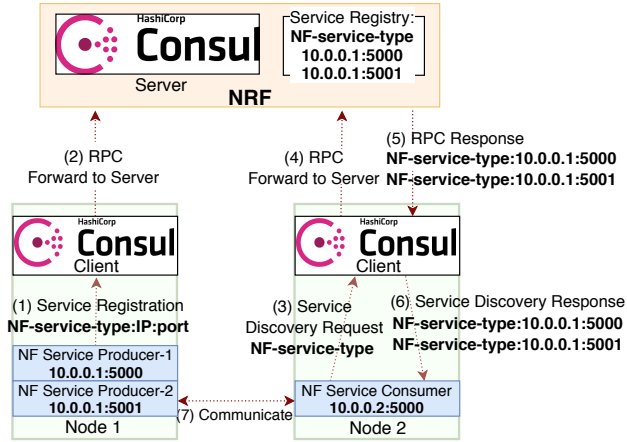


Figure 4.5: NRF Service Registration & Discovery Setup.

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.

4.4.3 Call Flow among the NFs

Figs. 4.6 & 4.7 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.

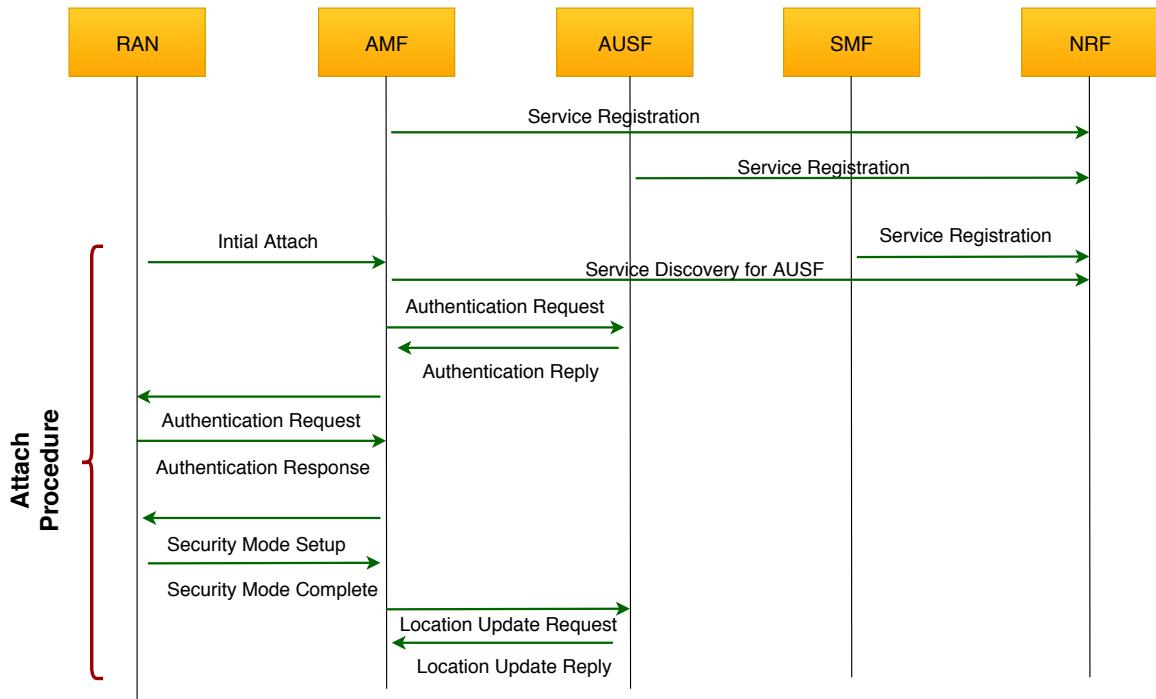


Figure 4.6: Call Flow Diagram (1).

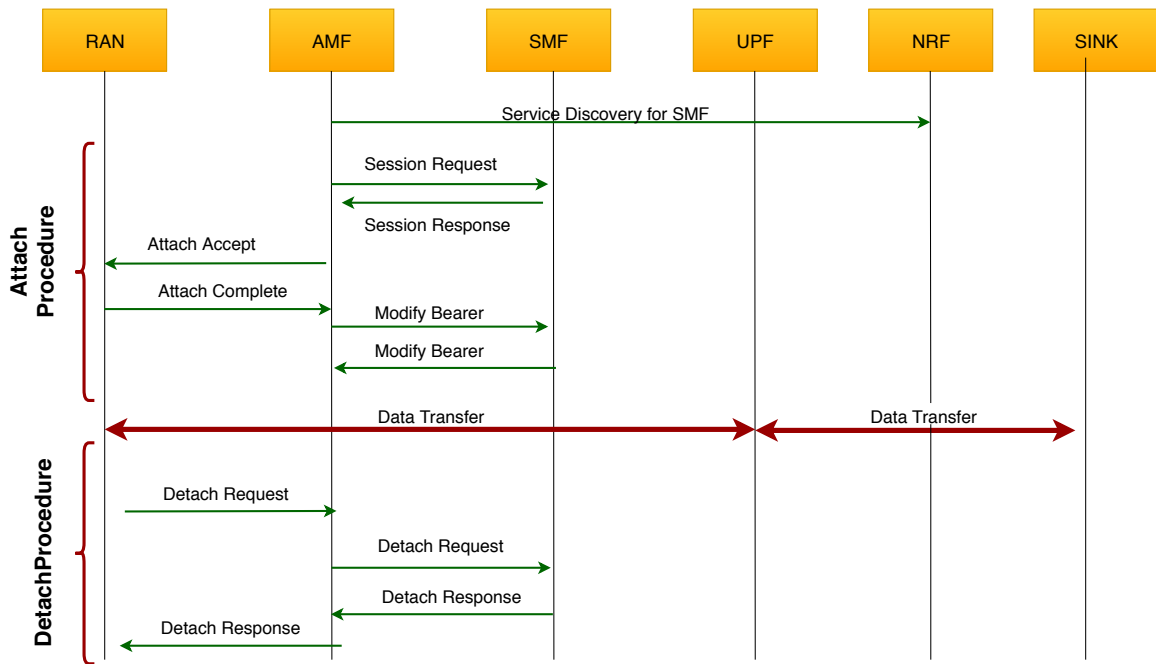


Figure 4.7: Call Flow Diagram (2).

4.4.4 Evaluation of Testbed

We evaluate our testbed by running the 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 concerning 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. 4.6 and Fig. 4.7) of attach procedure latency ($attach_l$) and detach procedure latency ($detach_l$).

$$CP_l = attach_l + detach_l \quad (4.1)$$

In this experiment, only a single instance of each of 5GC NFs 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, each NF can increase utilization of CPU cores based on processing requirement.

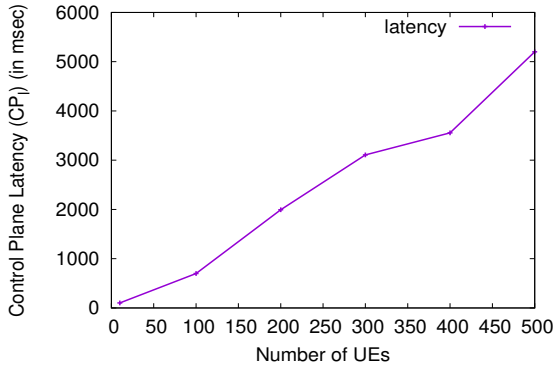


Figure 4.8: Variation of CP_l with UEs.

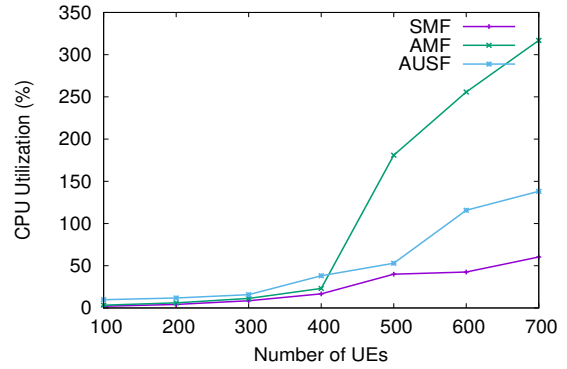


Figure 4.9: CPU Utilization of various NFs.

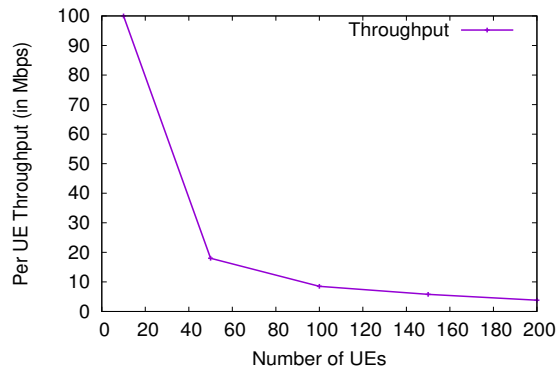


Figure 4.10: Individual UE throughputs with Concurrent UEs.

CP_l Measurement

From Fig. 4.8, 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.

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 several CPU cores. Hence we see the CPU utilization of NFs going higher than 100%. From Fig. 4.9, 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.

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. 4.10.

4.5 Load-Balancing Architecture for GRPC based 5G Core

From Figs. 4.8 & 4.9, 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.

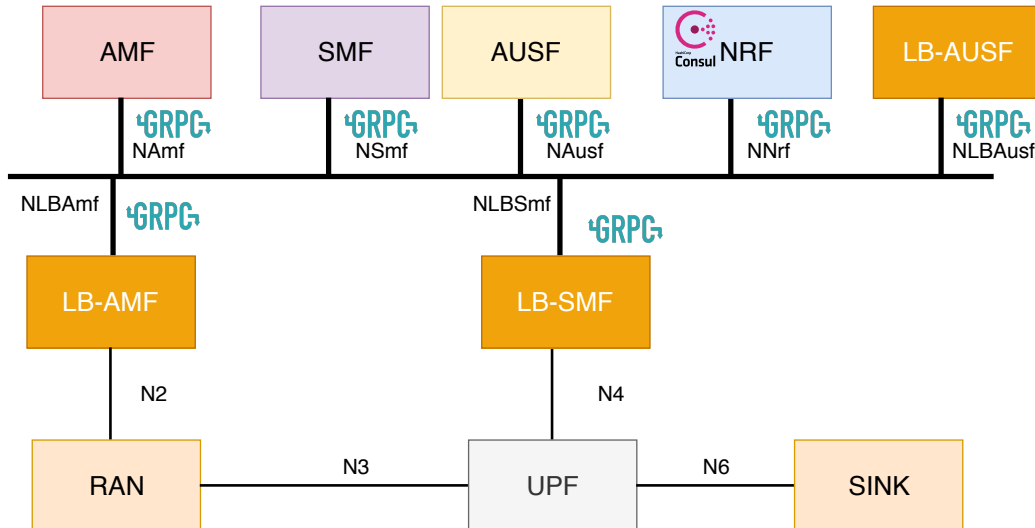


Figure 4.11: Load balancer SBIs in SBA-5G.

If there are multiple instances of each NF, many UEs can be handled concurrently and thereby

reduces 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. 4.11.

4.5.1 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 reports with the LB, and then 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* [26]. There is a particular LB server called the Look-Aside Load Balancer (LALB). The client queries the LALB, and the LALB implements load-balancing 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.

4.5.2 Implementation Framework

The implementation framework has been shown in Fig. 4.12. The implementation uses Consul along with Docker container platform for virtualizing the NFs. Prometheus [36] is used as a monitoring tool to monitor the NFs of 5GC. Prometheus is an open-source monitoring and alerting toolkit.

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 containers. There are 3 LALBs, one each for AMF, AUSF, and SMF. Every NF instance of each 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. 4.13 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. Then it 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.

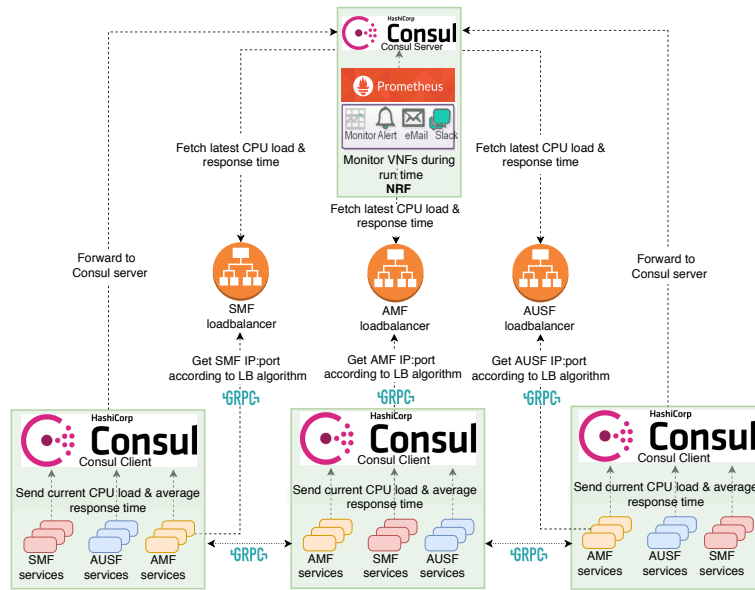


Figure 4.12: Implementation Framework.

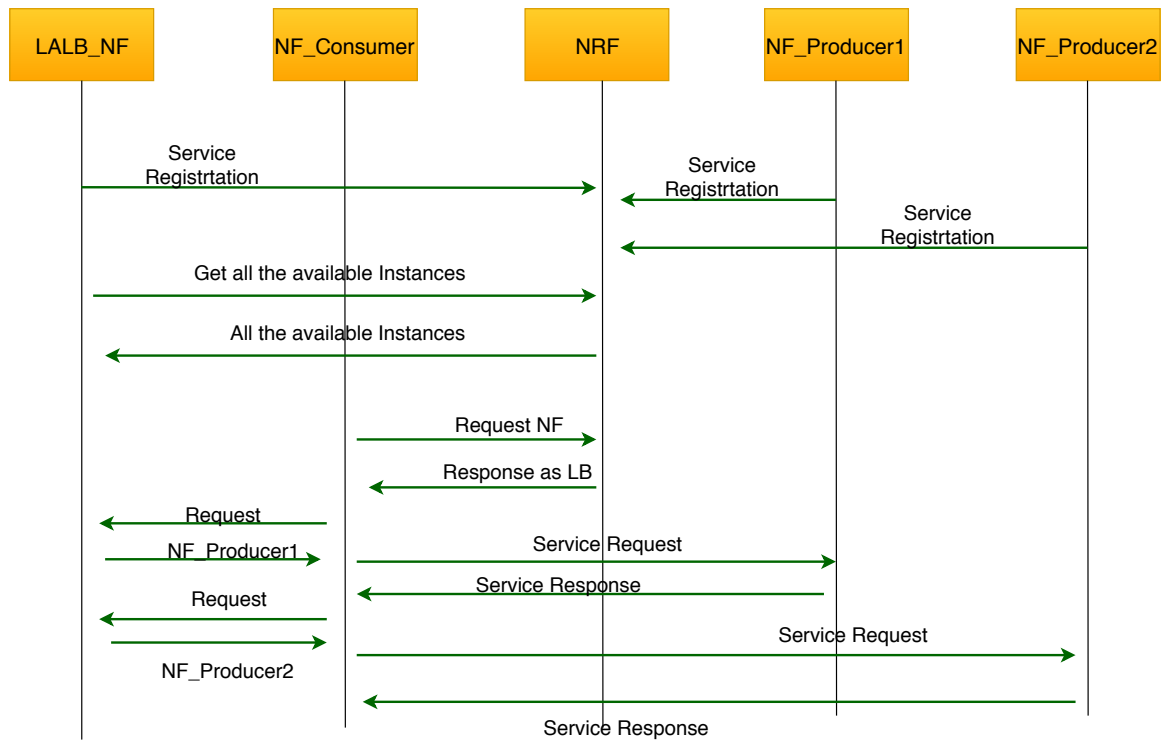


Figure 4.13: Load balancer interaction with NRF.

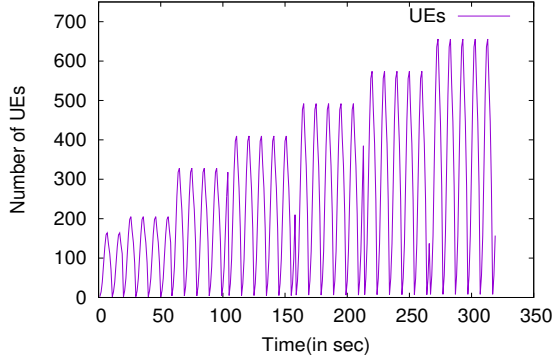


Figure 4.14: Variation of UEs with time.

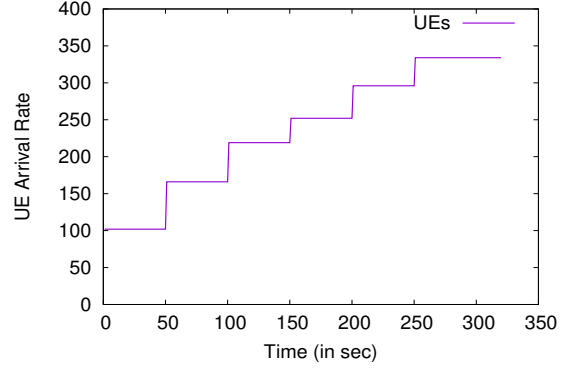


Figure 4.15: Average UE Arrival Rate with time.

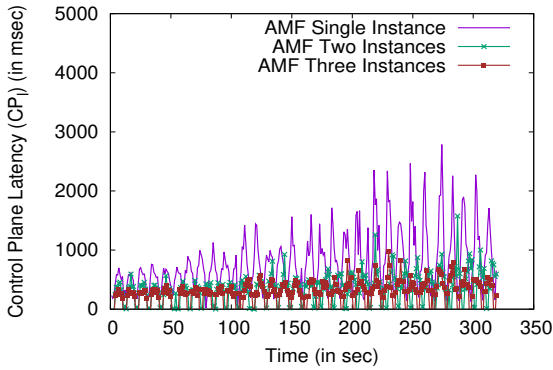


Figure 4.16: CP_l with time.

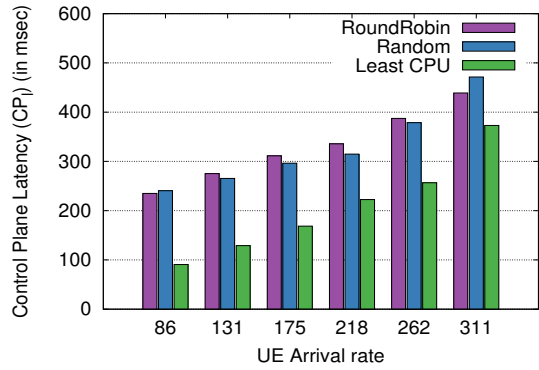


Figure 4.17: CP_l with Avg. UE Arrival Rate.

4.6 Evaluation of AMF LALB

In this section, we evaluate LALB on how effectively it can handle a large number of requests. For the evaluation, we are considering the load balancing of AMF NF because from Fig. 4.9, 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 as given in Table 4.2. 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.

4.6.1 Reduction of CP_l with increasing instances

Fig. 4.14 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. 4.15. Fig. 4.16 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

Table 4.2: 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

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.

4.6.2 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 concerning 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. 4.17, 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. 4.17 shows that CP_l for LCU is lesser than that of 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 therefore it 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 .

4.7 Summary

In this chapter, we have presented a model for the realization of SBI and SBA for 5GC and evaluated it concerning the control plane latency and resource utilization of various 5GC components and Dataplane throughput. We have proposed a 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.

Chapter 5

Conclusions and Future Work

To conclude, this thesis discusses various aspects related to prototyping and orchestration of network slicing in LTE and 5G core networks using open-source technologies. The network slices are created in the user plane of the core network and realized by differentiating the user traffic in terms of QoS. On top of the setup created, we explored various ways of orchestration mechanisms to ensure the SLAs of the users. The orchestration mechanisms include the horizontal and vertical scaling of the slices based on the type of network slice. The real-time monitoring of traffic in the slices is done. The slices of eMBB and mMTC are auto-scaled up or down based on the incoming load to the slice and current SLAs of the incoming flows.

The core network of 5G SBA is realized with the help of gRPC as SBI and Consul as NRF. On top of the setup created, we leveraged the concept of Look-aside Load Balancing to suit the 5G SBA and efficiently handle the incoming traffic load. We took the AMF as a use case for load balancing and conclude that the carefully chosen load balancing algorithms can significantly lessen the control plane latency when compared to simple random or round-robin schemes.

Some of the future works that we can derive from this thesis are mentioned below:

- The auto-scaling procedures that are applied to network slices in Chapter-3 can be studied and investigated in terms of performance using the concepts of time series analysis, control theory, reinforcement learning, and queuing theory.
- The uRLLC type network slice can be realized using multi-node setup. Since the traffic of uRLLC requires low latency, the scheduling of uRLLC flows has to be done by considering the end-to-end delay guarantees. The resource sharing of uRLLC type with other types and isolation can be studied.
- The candidate protocol stack for the realization of 5G SBI can be explored with various technologies to study the performance of the system.
- The admission control mechanism to the incoming users can be applied on the various network slices mentioned in Chapter-3, and also efficient resource management mechanisms among network slices can be studied.

Publications

1. **Tulja Vamshi Kiran Buyakar**, Anil Kumar Rangiseti, Antony Franklin A, and Bheemarjuna Reddy Tamma, “Auto-Scaling of Dataplane VNFs in 5G Networks”, in Proceedings of the 4th ManSDNNFV Workshop colocated with 13th IEEE International Conference on Network and Service Management (CNSM), November 2017.
2. **Tulja Vamshi Kiran Buyakar**, Amogh P.C, Bheemarjuna Reddy Tamma, and Antony Franklin A, “Poster: Scalable network slicing architecture for 5G”, in Proceedings of the 24th ACM Annual International Conference on Mobile Computing and Networking (MobiCom), November 2018.
3. **Tulja Vamshi Kiran Buyakar**, Harsh Agarwal, Bheemarjuna Reddy Tamma, and Antony Franklin A, “Prototyping and Load Balancing the Service Based Architecture of 5G Core using NFV”, in Proceedings of the 5th IEEE Conference on Network Softwarization (NetSoft), June 2019.

References

- [1] C. V. Forecast. Cisco visual networking index: Global mobile data traffic forecast update 2015-2020 2016.
- [2] 3GPP. 3GPP TS 23.501 - System Architecture for the 5G System 2018.
- [3] NFV-LTE-EPC. <https://github.com/networkedsystemsIITB> 2017.
- [4] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega et al. Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks. *IEEE Communications Magazine* 55, (2017) 72–79.
- [5] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira. Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges. *IEEE Communications Magazine* 55, (2017) 80–87.
- [6] V. K. Choyi, A. Abdel-Hamid, Y. Shah, S. Ferdi, and A. Brusilovsky. Network slice selection, assignment and routing within 5G Networks. In Proceedings of the IEEE Conference on Standards for Communications and Networking (CSCN). 2016 1–7.
- [7] A. Jain, N. Sadagopan, S. K. Lohani, and M. Vutukuru. A comparison of SDN and NFV for re-designing the LTE packet core. In Proceedings of 20th Conference on IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, 2016 74–80.
- [8] Linux. Linux Containers. <https://linuxcontainers.org> 2018.
- [9] R. S. Prasad, M. Jain, and C. Dovrolis. Socket buffer auto-sizing for high-performance data transfers. *Journal of GRID computing* 1, (2003) 361–376.
- [10] F. Z. Yousaf, M. Gramaglia, V. Friderikos, B. Gajic, D. von Hugo, B. Sayadi, V. Sciancalepore, and M. R. Crippa. Network slicing with flexible mobility and QoS/QoE support for 5G Networks. In Proceedings of the IEEE International Conference on Communications (ICC) Workshops. IEEE, 2017 1195–1201.
- [11] T. Taleb, A. Ksentini, and A. Kobbane. Lightweight mobile core networks for machine type communications. *IEEE Access* 2, (2014) 1128–1137.
- [12] NGMN Description of Network Slicing Concept. https://www.ngmn.org/fileadmin/user_upload/160113_Network_Slicing_v1_0.pdf 2017.

- [13] B. Chatras, U. S. T. Kwong, and N. Bihannic. NFV enabling network slicing for 5G. In Proceedings of the 20th Conference on Innovations in Clouds, Internet and Networks (ICIN). IEEE, 2017 219–225.
- [14] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina. Network slicing in 5G: Survey and challenges. *IEEE Communications Magazine* 55, (2017) 94–100.
- [15] Vertical Scaling. <https://searchcio.techtarget.com/definition/vertical-scalability>.
- [16] OpenStack. <https://www.openstack.org> 2018.
- [17] OpenBaton. <https://openbaton.github.io> 2017.
- [18] AMQP. <https://www.amqp.org> 2017.
- [19] RabbitMQ. <https://www.rabbitmq.com> 2017.
- [20] Docker. Docker. <https://www.docker.com> 2018.
- [21] Zabbix. <https://www.zabbix.com> 2017.
- [22] Open vSwitch. <https://www.openvswitch.org> 2018.
- [23] Quality of Service in Openstack. <https://docs.openstack.org/newton/networking-guide/config-qos.html> 2018.
- [24] Google. gRPC. <https://grpc.io> 2018.
- [25] HashiCorp. Consul. <https://consul.io> 2018.
- [26] Lookaside Load balancing in gRPC. <https://grpc.io/blog/loadbalancing> 2018.
- [27] NGMN. Service Based Architecture in 5G . https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180119_NGMN_Service_Based_Architecture_in_5G_v1.0.pdf 2018.
- [28] 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 Proceedings of the IEEE International Conference on Communications (ICC) Workshops. IEEE, 2018 1–6.
- [29] 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 (MobiCom). ACM, 2018 669–671.
- [30] C. Rotter, J. Illés, G. Nyíri, L. Farkas, G. Csatári, and G. Huszty. Telecom strategies for service discovery in microservice environments. In Proceedings of the 20th Conference on Innovations in Clouds, Internet and Networks (ICIN). IEEE, 2017 214–218.
- [31] 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 Proceedings of the IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). IEEE, 2018 1–7.

- [32] P. Amogh, G. Veeramachaneni, A. K. Rangiseti, B. R. Tamma, and A. A. Franklin. A cloud native solution for dynamic auto scaling of MME in LTE. In Proceedings of the IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). IEEE, 2017 1–7.
- [33] Google. Protocol Buffers. <https://developers.google.com/protocol-buffers> 2018.
- [34] ETSI. ETSI NFV Architecture. <https://www.etsi.org/technologies-clusters/technologies/nfv> 2017.
- [35] iperf3. <https://iperf.fr> 2017.
- [36] Prometheus. <https://prometheus.io> 2018.