DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

OPENEPC INTEGRATION WITHIN 5GTN AS AN NFV PROOF OF CONCEPT

Author	Muhammad Arif
Supervisor	Mika Ylianttila
Second Examiner	Mehdi Bennis
(Technical Advisor	Pasi Maliniemi)

February 2017

Arif Muhammad. (2017) OpenEPC Integration within 5GTN as an NFV Proof of Concept. University of Oulu, Degree Programme in Wireless Communications Engineering. Master's Thesis, 69 p.

ABSTRACT

Gone are the days, when a hardware is changed on every malfunctioning and the whole operation either stays down or load on the replacing hardware becomes too much which ultimately compromises the QoS. The IT industry is mature enough to tackle problems regarding scalability, space utilization, energy consumption, cost, agility and low availability. The expected throughput and network latency with 5G in the cellular Telecommunication Networks seems to be unachievable with the existing architecture and resources. Network Function Virtualization promises to merge IT and Telecommunications in such an efficient way that the expected results could be achieved no longer but sooner. The thesis work examines the compatibility and flexibility of a 3GPP virtual core network in a virtualization platform. The testbed is established on an LTE (Long Term Evolution) based network being already deployed and OpenEPC is added as virtual core network on it. The integration of OpenEPC in 5GTN (5TH Generation Test Network) is discussed in details in the thesis which will give an account of the possibility of implementing such a simulated vEPC (Virtual Evolved Packet Core) in a real network platform. The deployed setup is tested to check its feasibility and flexibility for a platform which could be used for NFV deployment in future. The monitoring of OpenEPC's individual components while utilizing the major resources within them, forms the primary performance test. The CPU Load and Memory Utilization is tested on different CPU stress levels having a constant data traffic from actual UEs. At the completion of the thesis work, a consensus is built up based on the test results that the test setup can hold number of subscribers to a certain amount without any performance degradation. Moreover, the virtual core network throughput and network latency is also compared to the commercial LTE networks and theoretical maximum values on similar resources to check performance consistency OpenEPC must offer.

Key words: OpenEPC, Scalability, Throughput, Network Latency, CPU Load, Memory Utilization, 5GTN, LTE

TABLE OF CONTENTS

FOREWORD

The thesis work is an effort to make a contribution in Mobile Wireless Communications research world, no matter how meagre and small it would be. I strongly believe in the words of 'Ralph Waldo Emerson' so I dedicate this work to all those esteemed hard working researchers who are spreading words of wisdom to the world out there.

"*Do not follow where the path may lead. Go, instead, where there is no path and leave a trail."* — Ralph Waldo Emerson.

The thesis work is accomplished at CWC, University of Oulu in the 5GTN project. The support and motivation from the entire 5GTN team made the work worth studying. I learned a lot of new things which enhanced my practical skills by strengthening the theoretical knowledge I already had. I find myself lucky to be part of the 5GTN architecture building team as such a testbed will help researchers in future to better understand and test their innovative ideas. I would like to thank especially Pasi Maliniemi, my technical advisor on the thesis, as his appreciation and guidance motivated me a lot.

The entire 5GTN project members helped and guided me during the whole journey. I would like to mention here Jaakko Leinonen, whose technical expertise helped me accomplishing the work. Thanks a lot for your time and guidance, you were always there for me when needed. I would like also to thank my supervisor Professor Mika Ylianttila for his guidance and suggestions at each stage.

Oulu, February 05, 2017

Muhammad Arif

LIST OF ABBREVIATIONS AND SYMBOLS

3GPP	Third-Generation Partnership Project
MNO	Mobile Network Operators
LTE	Long Term Evolution
MANO	Management and Network Orchestration
SDN	Software Defined Networking
PDN	Packet Data Network
5G	5th Generation
IFOM	IP Flow Mobility
MAPCON	Multi Access PDN Connectivity
IMS	Internet Multimedia Subsystem
MSC	Mobile Switching Centre
IoT	Internet of Things
TDD	Time Division Duplex
FDD	Frequency Division Duplex
vLAN	Virtual Local Area Network
IMSI	International Mobile Subscriber Identity
Hz	Hertz
MME	Mobility Management Entity
SGW	Serving Gateway
PDN GW	Packet Data Network Gateway
VPN	Virtual Private Network
VM	Virtual Machine
MEC	Mobile Edge Computing
MIMO	Multiple Input Multiple Output
MTC	Machine Type Communication
EPC	Evolved Packet Core
LTE-M	LTE-MTC
NB-IoT	Narrow Band IoT
PoC	Proof of Concept
CPU	Central Processing Unit
SPGW	Serving Packet Gateway
UE	User Equipment
RAN	Radio Access Network
L2	Layer 2
KVM	Kernel Based Virtual Machine
MAC	Media Access Control
VMNet	VM Network
NIC	Network Interface Control
COTS	Commercial off the Shelf
VNF	Virtual Network Functions
OS	Operating System

P-CSCF	Proxy Call Session Control Function
NFVI	Network Function Virtualization Infrastructure
S-CSCF	Serving Call Session Control Function
I-CSCF	Interrogating Call Session Control Function
PCRF	Policy and Charging Rules Function
HSS	Home Subscriber Server
IT	Information Technology
EMS	Element Management System
DNS	Domain Name Server
NAT	Network Address Translation
IP	Internet Protocol
ETSI	European Telecommunications Standards Institute
VIM	Virtual Infrastructure Manager
NFVO	NFV Orchestrator
CAPEX	Capital Expenditure
OPEX	Operating Expenditure
WCDMA	Wide Band Code Division Multiple Access
GSM	Global System for Mobile Communications
SAE	System Architecture Evolution
RNC	Radio Network Controller
GPRS	General Packet Radio Service
GGSN	Gateway GPRS Support Node
ANDSF	Access Network Discovery and Selection Function
QoS	Quality of Service
AAA	Authentication Authorization and Accounting
ePDG	Evolved Packet Data Gateway
ANGw	Access Network
UTRAN	Universal Terrestrial Radio Access Network
EUTRAN	Evolved UTRAN
GERAN	GSM EDGE Radio Access Network
NAS	Non-Access Stratum
RRC	Radio Resource Control
S1AP	S1 Application Protocol
SCTP	Stream Control Transmission Protocol
APN	Access Point Name
VoLTE	Voice over LTE
ProSe	Proximity Services
VNFM	Virtual Network Functions Manager
NFVSO	NFV Service Orchestrator
OSS	Operations Support System
BSS	Business Support System
NGMN	Next Generation Mobile Networks
SIM	Subscriber Identity Module

BSC	Base Station Controller
wlan	Wireless LAN
eth	Ethernet
LMA	Local Mobility Anchor
SNAPTR	Straight Forward Naming Authority Pointer
GTP	GPRS Tunneling Protocol
MAG	Mobile Access Gateway
BBERF	Bearer Binding and Event Reporting Function
AF	Application Functions
MDF	Media Delivery Functions
HTTP	Hyper Text Transmission Protocol
CDF	Cumulative Distributive Function
CGF	Charging Gateway Function
GUI	Graphical User Interface
PCEF	Policy and Charging Enforcement Function
DHCP	Dynamic Host Resolution Protocol
ARP	Address Resolution Protocol
PLMN	Public Land Mobile Network
MNC	Mobile Network Code
MCC	Mobile Country Code
NMSI	National Mobile Subscriber Identity
MSIN	Mobile subscriber identification Number
GUMMEI	Globally Unique MME identifier
TAC	Tracking Area Code
RAI	Routing Area Identity
TAI	Tracking Area Identity
GCS	Group Communications System
PCC	Policy and Charging Control
MBR	Maximum Bit Rate
GBR	Guaranteed Bit Rate
QCI	QoS Class Identifier
SRN	Shared Reference Network
JSON-RPC	Java Script Object Notation Remote Procedure Call
API	Application Programmable Interface
OPc	Operator Code
SPR	Subscriber Profile Repository
PMIP	Proxy Mobile IP
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
I/O	Input/output
POC	Proof of Concept
CPE	Customer Premise Equipment

5GTN	5G Test Network
CWC	Centre for Wireless Communications
SRN	Shared Reference Network
CND	Core Network Dynamics
5GTNF	5G Test Network Finland

1. INTRODUCTION

Network Function Virtualization being an emerging sensation in mobile networks has evolved to a great extend since its inception that MNOs have started already to shift their existing networks to it [1] . Primarily, virtualizing LTE core network elements to be used as virtual network functions makes it scalable and cost efficient [2]. The future next generation mobile networks are expected to have MANO with SDN capabilities where instead of mere Data and Control plane split, actual SDN implementation would be made possible. OpenEPC, a software implementation of LTE Evolved Packet Core, in compliance with 3GPP release 11 & 12 has come up with advanced unique features like Open Flow separation, IFOM and MAPCON which could be utilized as a 5G proof of concept testbed[1]. MSC with centralized IMS services, a single software framework for the whole network, all in one testing tools and Wifi to eNodeB emulation are some of the salient attributes of OpenEPC[1]. 5G test network at University of Oulu was keen to deploy OpenEPC as part of the network so this thesis will go through every bit of details from deployment to performance evaluation of OpenEPC regarding its implementation in 5GTN.

1.1. Background

This section will give a brief summary on 5G test network at University of Oulu and OpenEPC as its core network. It gives a brief account of 5GTN in terms of its RAN, spectrum licenses and the overall network.

1.1.1. 5G Test Network University of Oulu

5GTN aims to have deployed a fully functional LTE network which could be used as a 5G proof of concept testbed for future next generation mobile networks. Since 5G is a future technology which is expected to have the capability of supporting very high quality multimedia and cloud based services with less than 1ms network latency, the existing LTE architecture should be evolved efficiently. University of Oulu at CWC via 5GTN is striving to build a testbed architecture which could be used for hands on testing for research, business and innovative purposes. The core aim of the 5GTN project is to provide a best and appealing yet effective 5G test network ecosystem for research and business development purposes. 5GTN is part of the 5GTNF project (funded by Tekes), functional under the joint collaboration of representations from both industry and academia. Global manufacturers like Nokia, Ericsson, Huawei, Coriant and Intel along with internationally recognized research organizations like CWC, VTT, University of Helsinki, Aalto University and Tampere University of Technology altogether forms the research consortium. 5GTN has an open and scalable network at the University of Oulu and a restricted network at VTT. 5GTN forms a complete network where some network functions are shared and some are managed independently. VTT is a primary choice for network management, QoS optimization and network monitoring related use cases. University research focus is on wireless

[1] OpenEPC by CND. For details please visit http://www.openepc.com/home/overview/

interfaces, radio access architectures and related innovations. University will offer 5GTN for partners primarily via open interfaces assuming open network functionalities. The testbed also has connection at the test lab in Oulu University hospital primarily for medical IoT use cases. 5GTN at CWC is currently using and have the licenses to 2.6GHz LTE band 7 and 3.4-3.6GHz LTE TDD band 42 which is to be used within University of Oulu premises for research purposes. The detailed network description can be summarized by analyzing Figure 1 which shows the three vLAN connections along with multiple core networks present in 5GTN.[2] [3]

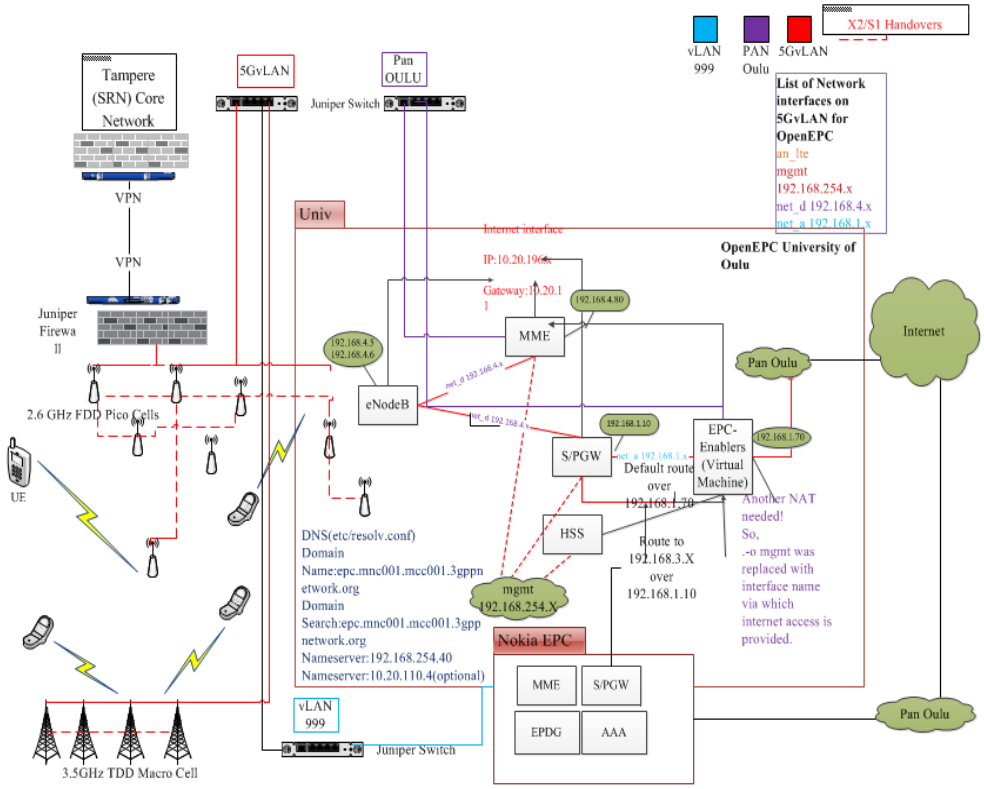


Figure 1. 5GTN overall architectural layout.

Nokia small cells are used for the LTE radio and we have multiple options when it comes to the core network. The Pico and Macro cells are commissioned primarily matching the configuration of the MME according to the Control and Data plane of the respective core networks so that cell knows to which core it has to connect to. First of all, an SRN core, based in Tampere owned by Nokia is connected to 5GTN via VPN over the internet. Than OpenEPC, a virtual core network by Core Network Dynamics (CND) on 5GvLAN deployed at University IT servers on VMWare ESXi vSphere environment which can be accessed and controlled remotely within University firewalls. Last but not the least we have Nokia EPC at the vLAN 999 which is under installation and will be up and running very soon. Figure 1 gives a detailed account of the whole 5G test network with all the vLAN and core network connection possibilities. PAN Oulu, a public internet access, gives internet outbreak to the OpenEPC individual VMs. Apparently as evident from Figure 1 that vLAN

[2] 5GTN Finland project page available at http://5gtnf.fi/
[3] 5GTN 5G Test Network Project Page available at http://5gtn.fi/

connections from PAN Oulu makes the core network less secure as it is from a public network. So, 5GTN has decided to use its own firewalls within the core network as well in the near future or the individual VMs will have firewalls in them.

The OpenEPC deployment as a core network is a step ahead towards the virtualization of the mobile core networks which is the basics to deploy NFV and to have open interfaces instead of conventional end to end proprietary hardware. Yet the servers to which the OpenEPC is deployed as a cloud setup is present locally at the University and access to which is done remotely from the 'Univ Domain' within the University of Oulu network. Users are provisioned with respect to IMSI which in the case of OpenEPC has multiple options i.e. either using the HSS console or the Web GUI with 5G test sim cards or CND test sim cards. Deploying an NFV platform to have support for SDN is one of the future goals of 5GTN. Apart from that keeping in view the 3GPP release 13 for LTE-M and NB-IoT, the testbed could be used for integration with other base stations and gateways so as to process the data collected from different IoT sensor nodes to network servers on the internet. Speaking of 5G, 5GTN is working to have all the functionalities expected to be functional in 5G within the testbed like MEC, Massive MIMO, 5G radio proof of concept, Massive MTC, NFV/SDN[2].

1.1.2. OpenEPC as a Core Network

OpenEPC is a 3GPP software based virtual implementation of LTE core network designed by Core Network Dynamics. It has a flexible and powerful software framework coded in 'C' which is deployed over standard Linux environment. Having a modular design all protocols, interfaces and functionalities are placed in individual modules with a support for inter-module communication[12]. OpenEPC is up and running within 5GTN on local University VMWare ESXi servers and is integrated to be used out of the box as one of the primary core networks within the whole 5GTN. The VMWare vSphere hypervisor installed on University servers is a type 1 bare metal hypervisor which provides an abstraction between the server hardware and the applications running on top of it which is OpenEPC in this case. The network nodes are running as virtual instances having Ubuntu as guest operating system and the network topology at each node is constructed keeping in view the interfaces which OpenEPC uses by default. The SVN Code Repository of OpenEPC for University of Oulu has all the setups saved in it.

Integrating OpenEPC with existing 5GTN and having a performance evaluation to check the feasibility and scalability of the system is the primary purpose of this thesis work. OpenEPC as a core network will be discussed in details in the later sections of Chapter 3.

1.2. Objectives and Scope

The thesis work aims to have an analytical overview of implementing OpenEPC as a virtual core network in real network infrastructure. The scope of the thesis work is to check the feasibility of a virtual core network i.e. OpenEPC and to cope up with the possible bottlenecks which comes with virtualization. Different network performance

tests along with the CPU Load and Memory Utilization are taken within OpenEPC individual nodes. The objectives of the thesis work are to deploy OpenEPC and test the system within a virtualization platform. The idea is to setup such a testbed for future mobile networks which could be used out of box by researchers and software developers to test the functionality with their innovative ideas. At the end of the day, OpenEPC should work as a scalable core network supporting different use cases, be it IoT, SDN/NFV etc. related. So, the research questions to be answered in the thesis are:

RQ1: How virtualizing the core network elements affects its performance?

The thesis includes different network tests primarily measuring throughput and comparing it to the theoretical maximum. This research question is analyzed by having an observation on the test results and a comparison of them to MNOs maximum throughput on similar resources.

RQ2: How effective is the current deployed setup in terms of different performance measurements? If throughput is expected to be on a certain level, then does the CPU can withhold such a load on the Gateways?

The amount of CPU and Memory utilization in OpenEPC components while the interfaces are being loaded with users' data load, can affect the performance. The problem is approached by generalizing behavior of OpenEPC in terms of CPU load and Memory Utilization. The pattern is studied by increasing and decreasing number of terminals which in turn increases and decreases the CPU load. A logical conclusion corresponding to relationship between the increment of terminals and Throughput, CPU Load, and Memory Utilization is drawn out of the test results.

RQ3: How many subscribers can OpenEPC hold with current resources?

One of the aim of this thesis work is to check the capability of OpenEPC as a core network and as an NFV deployment platform. The maximum number of subscribers which an OpenEPC system can withhold without any performance degradation is finalized by adopting an analytical approach over the test results. Since completely loading the gateway with user data traffic is not possible so an Open Source tool is used which can load the vCPUs in the OpenEPC VMs. The CPU utilization at different levels along with the actual load is studied to find an appropriate answer to this research question.

1.3. Research Methodology

Having a pragmatic approach, the thesis uses empirical methods for the research conducted. The capability of OpenEPC, as a main stream core network in a platform which would be used in future as a 5G testbed architecture for research purposes, is being tested. The tests were conducted to come up with a general relationship between the user data traffic increment on the Data plane i.e. SPGW and Throughput, CPU Load and Memory Utilization. The interfaces are loaded with an Open Source tool along with varying the number of UE terminals to check the CPU Load on the Control and Data plane. An analytical approach is adopted to somehow put the Load tests into a logical conclusion. Since, complete loading of the CPU resources by an actual user traffic is not possible so the empirical statistics, taken from the load tests are analyzed to check if there is any logical pattern. Based on the results an inference is developed that with the successive increment of terminals and the Load generated by them, the CPU will be loaded to a particular value. Using similar criteria, the memory being

utilized by increasing the traffic load on the gateways is analyzed. Apart from that different network performance measuring tools are utilized for Throughput and Latency measurements. The results and findings then provides a conclusion on the capability of the setup to be used as an NFV deployment platform in the near future.

1.4. Thesis Structure

The thesis is structured in a way that the reader gets familiar with all the requisite knowledge beforehand getting to the actual measurements and performance evaluation section. Chapter 2 gives an insight to Virtualization in general and NFV in particular. It provides a detailed account of the basics of virtualization and the challenges which comes with NFV deployment. Chapter 3 has two major sections, the first section comprises of System Architecture Evolution and the existing mobile core networks, the later section provides with details regarding virtualization of core networks. OpenEPC, a software implementation of 3GPP EPC, is discussed in details in this chapter along with the possible deployment of it in a NFV platform in future. Chapter 4 forms the primary aim of the thesis work which in practice is the integration of OpenEPC as one of the core networks in 5GTN being deployed at University of Oulu, CWC. This chapter will give a detailed overview of the existing 5GTN initially and then the implementation of OpenEPC on a virtualized platform. Chapter 5 features performance evaluation of the 5G testbed to check the feasibility of the deployed setup and the way virtualized core network i.e. OpenEPC behaves with actual LTE RAN.

2. NETWORK VIRTUALIZATION

Network Virtualization is the detachment of the application and operating system from the underlying hardware in such a way that the network endpoints get abstracted from the physical arrangement of the network. [3] It is in fact simulation of a hardware platform in software[4] where the idea is to decouple the physical topology from the logical topology regardless of where they reside within the data center. The functionality of a physical hardware is transformed into a virtual instance usually termed as virtual machines which has the ability to function like a conventional hardware [3].

A network interface card in the host is primarily the starting point of computer networks, which is then connected to layer 2 segments. An L2 network which is one subnet in an L3 network is then formed using switches to have an interconnection between L2 network segments. Multiple L3 networks when connected together by routers, forms the Internet. L2 and L3 collectively forms a data center and switches make the interconnectivity of data centers possible. When it comes to virtual networks each of these components needs to be virtualized. [4]

In order to communicate, each computer system needs at least one NIC. With virtualization, running multiple VMs on a computer system is possible where each VM needs its own virtual NIC. This is done by a hypervisor e.g. VMWare that provides processor virtualization which implements as many virtual NICs as needed. The vNICs are interconnected by virtual switch which is connected to the physical NIC. [4]

2.1. Fundamentals of Virtualization

This section will give a brief overview on all the basic information requisite to understand the virtualization technology in general and Network Functions Virtualization in particular.

2.1.1. Virtual Machine

A virtual machine is a software instance that runs an operating system and applications like a traditional physical computer does. The hypervisor basically provides a platform to run virtual machines and makes the consolidation of computing resources possible. Each virtual machine contains its own virtual resources i.e. a virtual CPU, memory, hard disk, and network interface card.[5] VMs could easily be cloned, suspended, resumed and moved [3]. In a virtualized environment, a virtual machine is often referred as a guest operating system.

---

[4]VMWare Documentation available online at the link :https://pubs.vmware.com/vsphere-51/index.jsp?topic=%2Fcom.vmware.vsphere.vcenterhost.doc%2FGUID-ED375B12-7D08-4B7E-81EE-DCE83E51B1AF.html

[5]The information is available online and could be accessed at https://www.sdx-central.com/sdn/networkvirtualization/definitions/whats-network-virtualization/

2.1.2. Hypervisor

Hypervisor provides an abstraction layer between the physical hardware and the applications running on an operating system on top of that hardware. It makes virtualization possible by allowing hardware devices to share their resources between VMs running as guests and lying on top of the physical hardware [3]. The way a hypervisor functions depends upon the type of it like it acts as an operating system for operating systems and also as a virtual machine monitor.

2.1.2.1. Types of Hypervisors

Type -1 native or bare-metal Hypervisor is installed directly on a hardware so as to let it operate on top of the host's hardware. This type of hypervisor is used mostly in servers where the hypervisor runs directly on the server hardware without any operating system[4]. The server where the hypervisor runs services and different applications can then be accessed remotely within the network.

Type-2 Hosted Hypervisor is installed on the host operating system which provides an abstraction between the underlying hardware and the operating system along with applications which would then be installed on top of the hypervisor. The operating system and applications running within the hypervisor does not depend on the resources of the physical hardware. Type-1 hypervisor is used as an operating system directly installed on a server hardware where as Type2 hypervisor could be installed on a native operating system which is the primary distinction between the two as depicted in Figure 2. A good example of them would be VMWare vSphere ESXi and VMWare Workstation respectively.

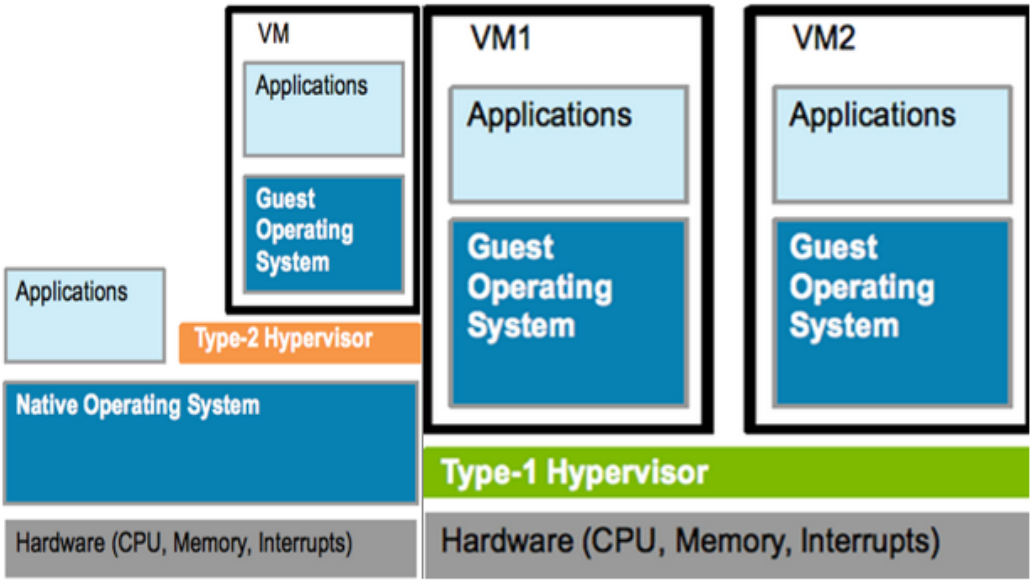


Figure 2. Hypervisor Categories. [4]

Since there are several vendors offering a variety of Hypervisors of which KVM, Xen and VMWare are some of the stakeholders. VMWare ESXi is the hypervisor used in the OpenEPC implementation in the 5G testbed being deployed locally at the

University of Oulu. ESXi which is also referred as vSphere is a bare-metal type 1 hypervisor being installed directly on a physical server hardware and the virtual machines are created inside the ESXi environment which may have different operating systems, running variety of applications. The detailed architectural description in Figure 3 would help us better understand the way this hypervisor works. The VMs run as virtual instance on the VMWare ESXi environment which is an OS and hypervisor at the same time. There could be multiple server host in a cluster which could also be interconnected. The server can then be remotely accessed anywhere in the network from a web client or VMWare remote connection console, if no firewall blocks the connection.

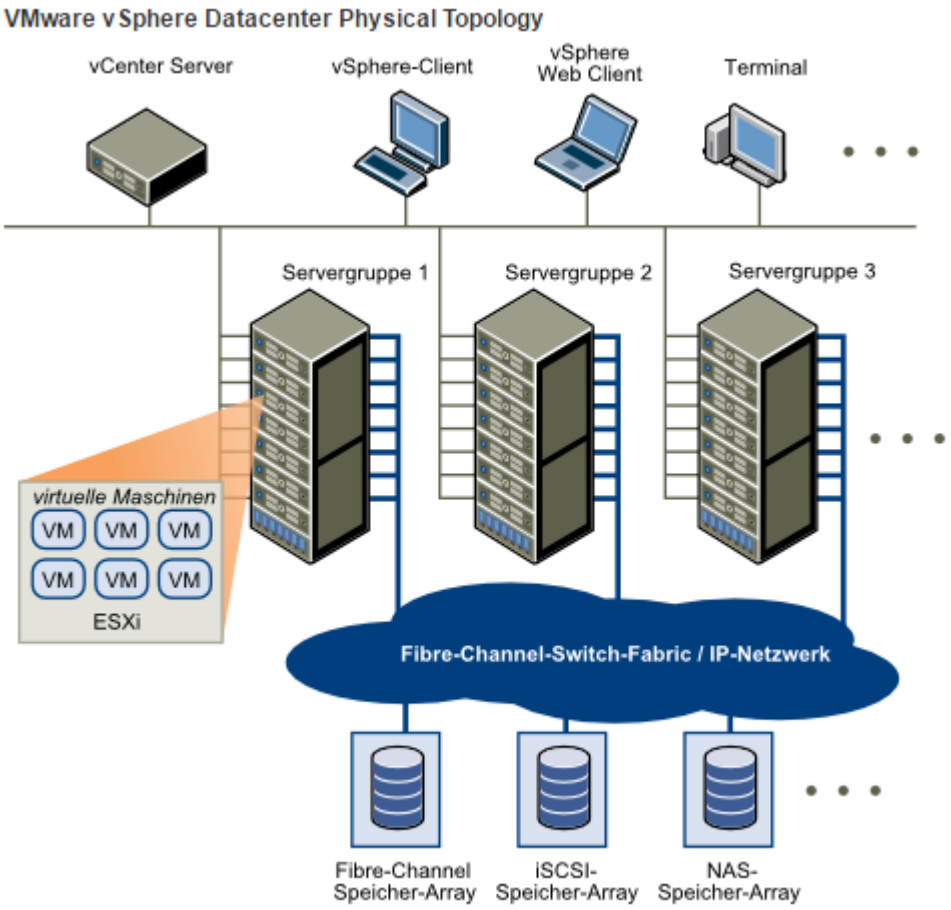


Figure 3. VMWare vSphere ESXi. [4]

2.1.3. Virtualizing a Network Card

Since, a virtualization platform primarily consists of host and guest machines. The host machine has a physical hardware on which guest machines run as virtual instances in different virtual machines. The hypervisor apart from being an operating system itself also provides a virtualization layer for the abstraction of host and guest machines. The network cards inside the guest machines is referred as virtual NIC which is either bridged to a physical NIC on the host or it can be connected to a virtual network created on the host [5]. The virtual NIC has its own unique MAC address and is similar to a

PCI Ethernet Controller in practice. VMWare VMNet driver loaded in the host operating system makes the Host-Guest connection possible. A virtual machine with a bridge virtual NIC connected to a virtual network does not require an Ethernet interface on the host and can fully participate in accessing and providing network services [5]. A VMNet driver installed on the host OS implements a virtual bridge [5] as depicted in Figure 4 to connect the physical NIC to that of the vNICs.

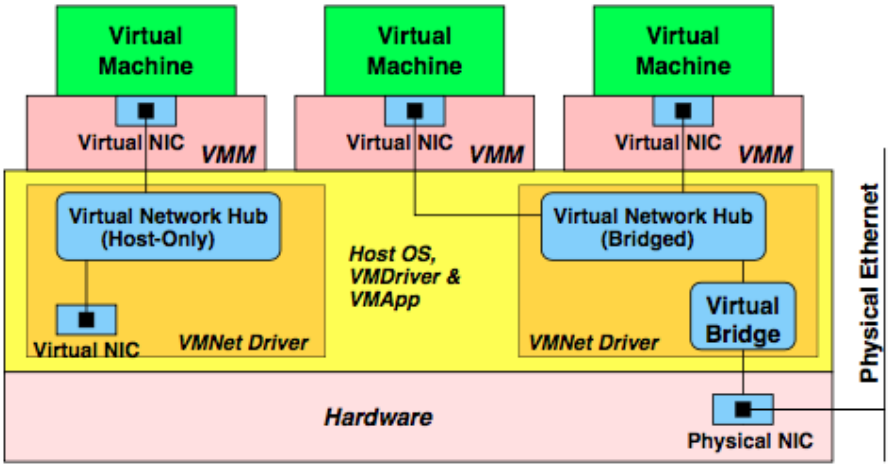


Figure 4. VMware Network Subsystem. [5]

2.2. Network Functions Virtualization

Since the network dependency on hardware resources has always been a limiting factor, service providers have always been striving hard to overcome the problem. Keeping in view the evolution of IT virtualization, service providers have been trying to increase life span of hardware as they become obsolete once the technology itself evolves. Network Function Virtualization, the transfer of proprietary hardware to software running on COTS equipment, has been identified as a potential solution to the problems MNOs are facing [6] , [2]. NFV provides an abstraction of the network functions from the underlying hardware by having a hypervisor layer in between, the hypervisor layer is also referred as 'virtualization layer'. The network functions will be virtual instance running as virtual machines on a server hardware with a hypervisor as an operating system. Apart from fulfilling the requirements for resource allocation, dependencies and other attributes, the framework holding the VNFs should have dynamic initiation and orchestration support [2].

2.2.1. Is NFV SDN

SDN and NFV are most often perceived as a similar technology and there has always been confusion in understanding both as completely different technologies. SDN and NFV are two different independent technologies yet they complement each other [2]. Since virtualization alone cannot solve service deployment problems and if put into unskilled hands can result in disastrous results so SDN with orchestration capabilities comes into practice reducing the risk to a greater extent [7]. SDN is a component of

NFV [7] and it serves NFV by providing the programmable connectivity between the VNFs [2]. The connection will be managed by orchestrator of the VNF which would be an SDN controller whereas NFV serves SDN by implementing core network functions [2]in software instances running on a standard server as host machine. NFV virtualizes the SDN controller to run on a cloud which could be migrated to different locations according to requirements on the network [2].

2.2.2. NFV in the Mobile Core

The virtualization and application of NFV to the 3GPP Evolved packet core forms the use case 5 according to ETSI NFV use cases[6]. The use case primarily aims to apply virtualization to the EPC, IMS and all the network functions like P-CSCF, S-CSCF, HSS, and PCRF etc. All the core network components once part of the NFV framework would be VNFs running on virtual machines which could then be orchestrated by an SDN based controller. One of the core reason for such a deployment is the scalability a virtual network has to offer as current mobile networks lack automatic network orchestration. NFV in the LTE mobile core will reduce the complexity of the network and would be cost efficient as the IT virtualization provides such a platform that expensive proprietary hardware is no longer needed. The tackling approaches of the control and data plane will definitely be different as VNFs in the data path will require different virtual resources as compared to the VNFs associated with signaling only[6]. NFV is also expected to provide such a scalable platform in terms of load balancing which is another crucial issue in mobile networks such that if any VNF is enough loaded to perform efficiently, the required component is created, deployed and configured automatically according to the requirements.

2.2.3. NFV Architectural Framework

The NFV architectural framework comprises of functional blocks and reference points between blocks. NFVI, VNF and MANO are the primary functional blocks. Figure 5 depicts ETSI NFV framework, where the solid lines represents main references and execution points. The dotted lines are the secondary focuses which are not so important at this stage. All the functionalities in Figure 5 are requisite for virtualization.[7]

The virtualization of a network function in a legacy non-virtualized network is called a virtualize network function (VNF) e.g. MME, SGW and PDN GW of a 3GPP LTE [2]. A VNF can be deployed in a single VM or it could be deployed over multiple VMs having multiple internal components. The deployment of a 3GPP virtual core network on a host machine separated by a hypervisor is a practical example of VNFs. The Element management system (EMS) will then manage the functionalities of a VM or within VM[7]. Every individual component in this case will be running on a separate VM with an operating system of its own. They are then interconnected by virtual interfaces and having a common DNS server to resolve IP queries within the network.

[6] ETSI GS NFV 001 v1.1.1 (2013–10)
[7] ETSI GS NFV 002 v1.1.1 (2013–10)

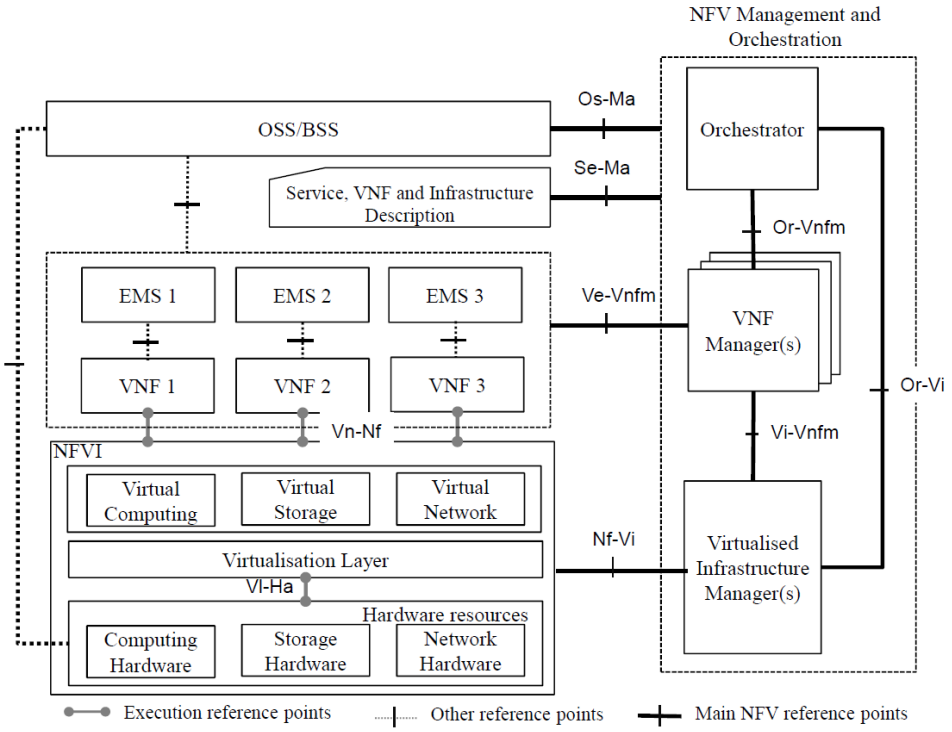


Figure 5. NFV reference architectural framework.[7]

The NFV Infrastructure as depicted in Figure 5 comprises of all the hardware and software resources used by the VNF including the hypervisor providing the virtualization layer. The physical hardware resources consist of network, storage and computing for connectivity, processing and storage respectively[7]. The VNF Manager, Virtualized Infrastructure Manager and the Orchestrator forms the Management and Orchestration of the NFV which is commonly referred to as MANO. The VNF manager being either deployed in every single VM or for multiple VMs, deals with the life cycle management of VNF which primarily includes instantiations, updates and termination etc. The orchestrator controls and manages the NFVI but the VIM has the task to manage the interaction of a VNF with the computing, storage and network resources. [7] [6] [2] The reference points which interconnects different functional blocks are described in Table 1 in details.

Table 1. Reference points in NFV Architecture[7]

Reference Points	**Interconnections**	**Description**
VI-Ha	virtualization layer-Hardware Resources	Interface between the virtualization layer and hardware resources to provide an execution environment for VNFs.
Vn-Nf	VNF-NFVI	This interface represents execution environment the NFVI provides to the VNF.
Or-Vnfm	Orchestrator-VNF Manager	Sends configuration information to the VNF manager for an appropriate configuration of VNFs. The VNF manager sends resource related requests using this reference point.
Vi-Vnfm	VIM-VNF Manager	The VNF manager entertains queries pertaining to resource allocation requests. It offers exchange of information related to virtualized hardware resource configuration and state information.

Or-VI	Orchestrator-VIM	The orchestrator requests resource allocation and reservation to configure hardware resources and state information.
Nf-Vi	VFVI-VIM	Keeping in view the resource allocation requests, it undergoes assignment of virtual resources along with the forwarding of the state information.
Os-Ma	OSS/BSS-MANO	This reference point requests for the management of network service and VNF lifecycle which also includes policy management and data analytic exchanges.

2.2.4. Management and Network Orchestration

MANO features the primary task of managing and orchestrating a virtual network, in NFV implementation. The ETSI MANO consists of three functional blocks as shown in Figure 5 i.e. VIM, NFVO, VNFM and data repositories [6]. The virtualized infrastructure manager (VIM) has the task of controlling both the physical and virtual network functions infrastructure [6]. Every VNF has a manager called as VNFM which has to manage the smooth running of the associated VNFs. It is important to note that a VNFM can control and manage either a single VNF or multiple VNFs could be assigned to a single VNFM. NFVO being capable of both service and resource orchestration has a major role of orchestration [7] [2]. It ensures an end to end service as part of service orchestration by creating VNFs and managing the network topology of them [7]. Finally, the data repositories which are four in number serves as data bases which holds information related to availability, creation, deployment and allocation of resources[7] [6].

2.3. Benefits of Virtualization

Network virtualization has ample of benefits from both business and technical perspective. Since the commercial proprietary hardware is usually very expensive so deploying a conventional network always costs a lot. These network components if virtualized will reduce both CAPEX and OPEX as the expensive hardware equipment will no longer be used, instead virtual machines running on COTS software will be used. Apart from that, the expenses on maintenance and energy costs is reduced to be non-existent. There would definitely be no more space utilization once each component in a network goes virtual. The primary advantage of virtualization particularly in mobile core networks is the flexibility offered by it. The components can easily be scaled according to network demands and could also be shifted to different locations. In fact, it would be nothing different than a copy paste operation. This helps in dealing with load balancing where a component if loaded too much could be assigned more virtual resources.[8] [9]

[8] Virtualization: Benefits and Challenges An *ISACA Emerging Technology White paper Available at https://www.thedatachain.com-materials/virtulization_wp_27oct2010_research.pdf*

[9] Christine, Leja, Richard C. Barnier, Charles L. Brown, Paul F. Dittmann, Paul Koziel, Mark Welle, J.T. Westermeier, "Virtualization and Its Benefits", *White Paper October 14,2008 By AITP Research and Strategy Advisory Group*. https://www.aitp.org.

3. EVOLVED PACKET CORE OVERVIEW

This chapter will give a detailed overview on 3GPP Evolved Packet Core in the first section and then the expected changes in its architecture with the next generation mobile networks in later sections. The virtualization of LTE core network is than discussed keeping in view the expected outcomes. OpenEPC forms the major discussion in the later section of this chapter which is explained in details.

3.1. System Architecture Evolution

3GPP in 2004 proposed TCP/IP protocol for next generation core and defined IP-based flat architecture as a part of SAE. It is in compliance with 3GPP according to TS 23.401 and TS 23.402 specifications for 3GPP and non-3GPP access respectively[10] [11]. The LTE/SAE architecture is designed to support high data transmission and high response rates. It is an evolution of existing GSM/WCDMA packet core network, with more simplified operations. It also includes IPv6 functionality with optimal traffic routing to have a flexible and scalable service provisioning. [8] Contrary to legacy networks, LTE/SAE architecture has no RNC but an eNodeB having connections with the other eNodeB base stations over X2 interface. It has an S1 protocol stack enabling communications between the RAN and the LTE core network i.e. EPC. The Circuit switching functionality no more exists in the core yet it has GGSN Node with the same role as in GPRS network [8], [15]. The functional elements inside EPC can be seen in Figure 6 where both control and data plane is shown. The PDN Gateway manages IP services such as traffic routing, addressing, security management, access to non-3GPP networks, and MME manages signaling traffic, authentication, authorization and mobility [8].

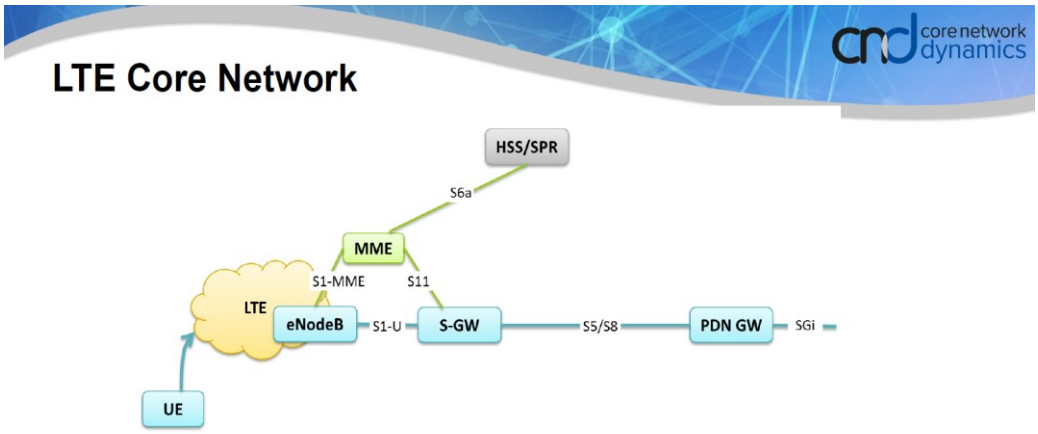


Figure 6. LTE core Network Basic Diagram.[12]

[10] TS 23.401 –General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network ;(E-UTRAN access).
[11] TS 23.402-Architecture Enhancements for non-3GPP Access.
[12] Wiki and Svn repository: OpenEPC for University of Oulu developed by CND core network dynamics. Training slides on OpenEPC by CND held at University of Oulu in week 42, 2016.

The SGW being part of both the Control and Data plane is the local mobility anchor. The HSS is the user subscriptions profile where all the authentication is performed for 3GPP access networks. Figure 7 below gives a detailed account of LTE EPC with QoS and Policy Control. The PCRF is the primary component involved in QoS which communicates with all the components over different interfaces such as SP, Gxx, Gx and Rx. The detailed architectural diagram in Figure 7 also depicts the client mobility support ensured by the ANDSF functionality over s14 interface. The EPC also has an integration support for 3GPP legacy networks such as UTRAN and GERAN. Moreover, both trusted and untrusted non-3GPP could also be integrated within the EPC using ANGW and ePDG respectively. S2a and S2b are the interfaces being used for the non-3GPP access. As far as the authentication and authorization of a client is concerned, the HSS and AAA server performs the task for 3GPP and non-3GPP respectively. SWm, Swa, and STa are the interfaces used by non-3GPP to get authenticated within the AAA server whereas the HSS uses S6a for LTE and S6d for 3GPP legacy networks. It is important to note that the HSS and AAA server also communicates over SWx interface.[12]

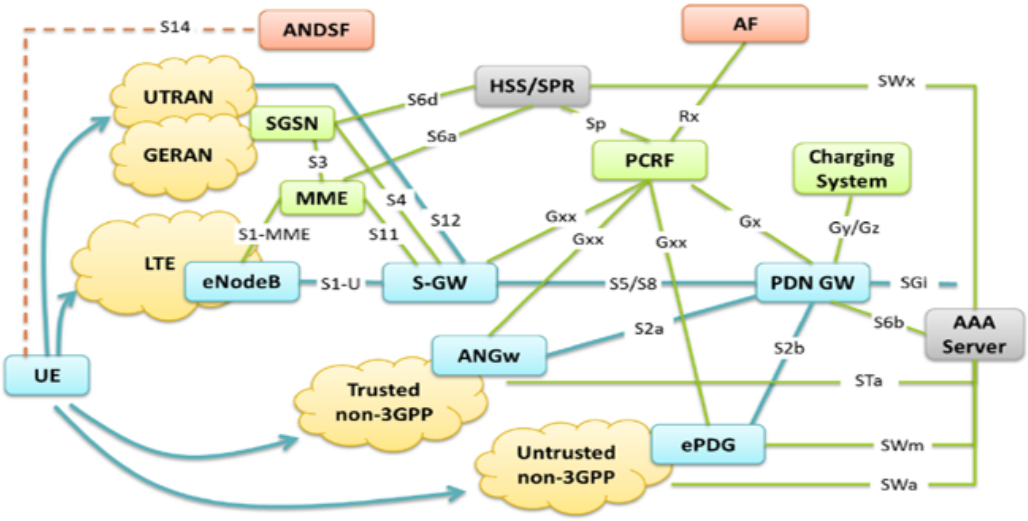


Figure 7. LTE EPC and RAN.[12]

3.2. Protocol Stacks in Control and Data Plane

Every component in LTE network including the RAN and UE has unique protocol stacks. Since the Control and Data plane is a separate entity so the protocol stack differs in both cases on every individual component in the LTE network. The Control plane path comprises of the UE, eNodeB and the MME. The protocol stack of the Control plane is illustrated in Figure 8. The LTE Radio protocol stack design has RRC stack which is responsible for broadcast of system information blocks by transporting dedicated information regarding NAS, UE capabilities etc. It controls paging, QoS, initial security activation etc. RRC having support for self-configuration and optimization controls the establishment and release of radio bearers and also take care

of the modifications in them. If there is a radio link failure, the RRC makes the recovery possible. [13]

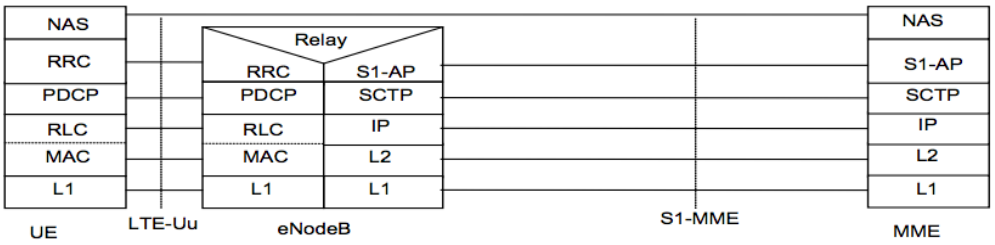


Figure 8. LTE Control Plane Protocol stack.[12]

The S1AP provides signaling between the eNodeB and the MME over the S1-MME interface. S1AP includes context management, Mobility and NAS transport as its primary functions. The IP packets are transported over SCTP protocol. [14] As part of the Control Plane stack, 'gtp' protocol stack is very imminent as depicted in Figure 9 which shows the protocols stacks used in the data plane of LTE network. The GPRS tunneling protocol makes the required data path tunnel between the PDN-gateway, the SGW and the eNodeB.

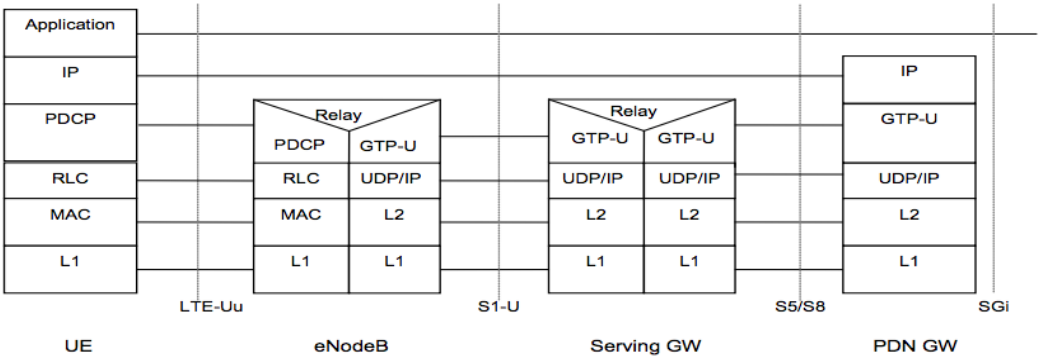


Figure 9. LTE Data Plane Protocol Stack.[12]

3.3. Attachment and Detachment Procedures for User Equipment

After having connected to LTE access network, a UE has to follow a series of steps to get its uplink and downlink ready. It is referred to as S1 setup owing to the S1AP protocol stack in the EPC and for the UE particularly the S1 messages are encapsulated in NAS at the eNodeB. Initially the eNodeB selects an MME based on the identities provided by either the UE or its own topology. The MME requests IMSI from the UE to perform NAS security setup which primarily includes integrity and ciphering. Since an APN configuration is added in the UE based on the APN profile within EPC, the MME selects PGN gateway using the corresponding APN name. The MME then selects Serving gateway to serve eNodeB. The attach request message contains ESM

[13] 3GPP TS 36.413- Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description.
[14] 3GPP TS 36.413- Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 Application protocol (S1AP)

PDN connectivity request message for default bearers. So, the MME creates the default bearer and send session request message for bearer creation in the SGW and PGW. Once the session is created a tunnel is created between the eNodeB and MME, the MME then provides it further to the SGW. The description in Figure 10 provides details of the procedure, from attach request till the session creation and successful attachment.[12]

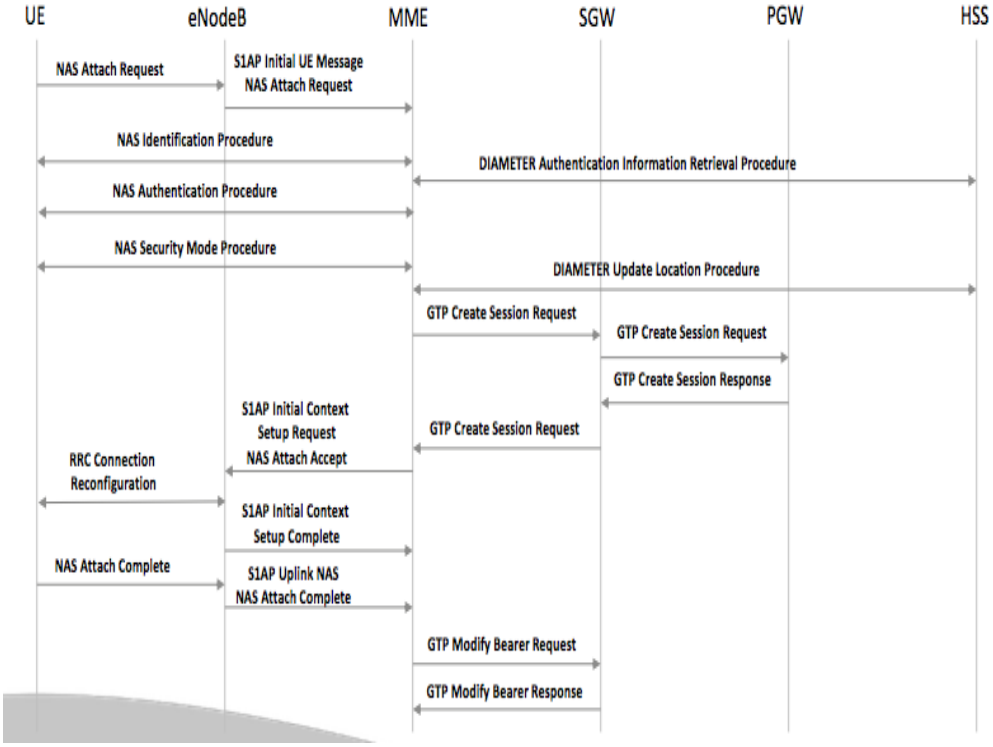


Figure 10. S1 setup procedure.[12]

As far as the detachment is concerned, the process is triggered either by the UE or the network. The bearers are removed in the gateways for all PDN connections. The steps followed by the UE to detach can be seen in Figure 11 below.

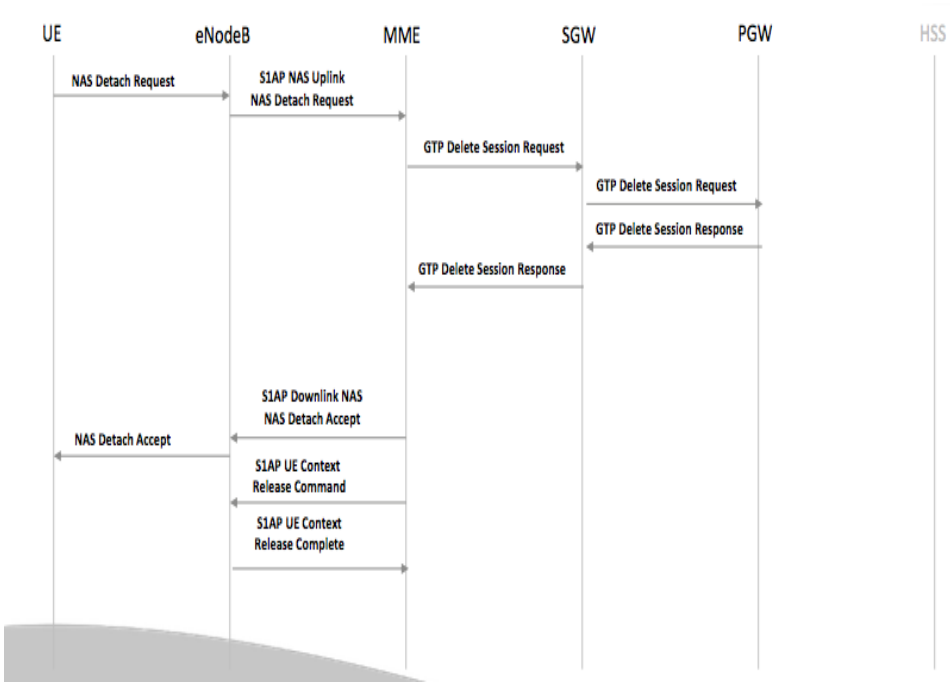


Figure 11. Subscriber Detachment. [12]

3.4. EPC Road Map To 5G

3GPP with release 8 introduced EPC as an all IP based new core network to LTE. Unlike the Legacy core networks, it has features like 'Policy and Charging' to ensure better QoS and ANDSF for a better access network selection. Emergency calls, multimedia broadcast and location services are added with release 9. Later on, it is complemented with more advanced features like MPCON and IFOM with release 10, with MTC in release 11, with VoLTE and ProSe in release 12.[15] Keeping in view the 5G use cases such as IoT, Virtual reality, augmented reality etc. the core network needs further evolution. Virtualizing core network functions to have a support for NFV deployment is the first step towards 5G to have more scalability. Though the current EPC stack supports splitting the control and data plane in the gateways but the software orchestration of the network with SDN tools is not possible as it lacks the required IP tunnels. Since SDN also introduces control plane functionality in a cloud which will then have several data planes in different remote locations so the new EPC would be quite different form the existing one within LTE. Moreover, contrary to service separation based on APN in existing core networks, network slicing in core network which is one of the features expected to be introduced in 5G will need orchestration to have management and control of all resources available within the network[15]. In short, 5G will alter the EPC into a low-cost core network which will be programmable, orchestrateable and cloud native with decomposed functionality[15].

[15] Diez, "Mobile Plots - From EPC to 5G", *Slideshare.net*, 2017. [Online]. Available: http://www.slideshare.net/AlbertoDiez4/mobile-plots-from-epc-to-5g.

3.4.1. Virtualizing Core Network Functions

5G core network elements will no longer be proprietary hardware boxes but it will be software instances running on virtual machines because virtualizing core network functions is deemed to be the basics of NFV deployment [2]. According to ETSI NFV, the EPC needs to be manageable and orchestrateable by VNFM and NFVSO respectively[7]. OpenEPC provides a virtual core network platform which is scalable and cost efficient. All the LTE core network elements are provided as virtual instances having COTS software components running as guest machines on top of host hardware which could be deployed in a network on VMWare, Dockers etc. There will be a hypervisor layer which gives an abstraction between the underlying hardware and the VNFs.

3.4.2. OpenEPC on NFV Platform

The components within OpenEPC are deployed as virtual machines on a virtualization platform which have a support for NFV deployment. Here we will discuss the scalability of OpenEPC in terms of Network Function Virtualization. Let's take for an example the HSS which is a primary authentication module within EPC-Enablers VNF, detects high load and the CPU utilization is reached to a maximum level. The EMS in this case will trigger an alarm and the OSS/BSS sends this load data to the NFVO. The Orchestrator after analyzing the situation makes a decision and send parameters asking for a new HSS to the VNF manager which along with the VIM then allocates new resources to make the component available. VNF manager then deploys the new HSS and configures it.

3.5. OpenEPC at a Glance

OpenEPC is an EPC prototype implemented in compliance with 3GPP [9]. OpenEPC is a prototype implementation of the 3GPP Release 11/12 Evolved Packet Core that will allow academic and industrial researchers around the world to obtain a practical look and feel of the capabilities of the Evolved Packet Core[1]. OpenEPC can be integrated with various access network technologies and different application domains and thus provides an excellent foundation for own research activities and the establishment of Next Generation Mobile Network (NGMN) testbeds [10]. It is a set of software components which offers advanced IP mobility schemes, QoS control and integration with different application platforms in network environments. OpenEPC as a testbed includes all technologies being deployed by the Core Network Dynamics to provide the EPC for research and deployment. [11] It is a platform offering an easy integration and emulation of the mobile networks from radio to the networks and applications which could be used out of the box. Integration of new wireless access technologies makes the OpenEPC unique in a sense that it gives new horizons for research when it comes to mobility, QoS, security and optimizations in the network architecture.

3.5.1. History of OpenEPC

Core Network Dynamics started the OpenEPC project back in 2008 as a research prototype which does not just cover the basic EPC features but it offers functional elements for the entire architecture.[1] There are several releases of which the latest is the OpenEPC Rel.7 containing all interfaces with different access technologies. It could be used as a mobile broadband testbed which works with base stations, modems and standard SIM cards[1]. In fact, CND is a company spin off from Fraunhofer FOKUS and OpenEPC project is accomplished within CND till release 7 in 2016. The time frame of OpenEPC since its inception is illustrated in Figure 12 below, along with the involvement of Fraunhofer FOKUS.

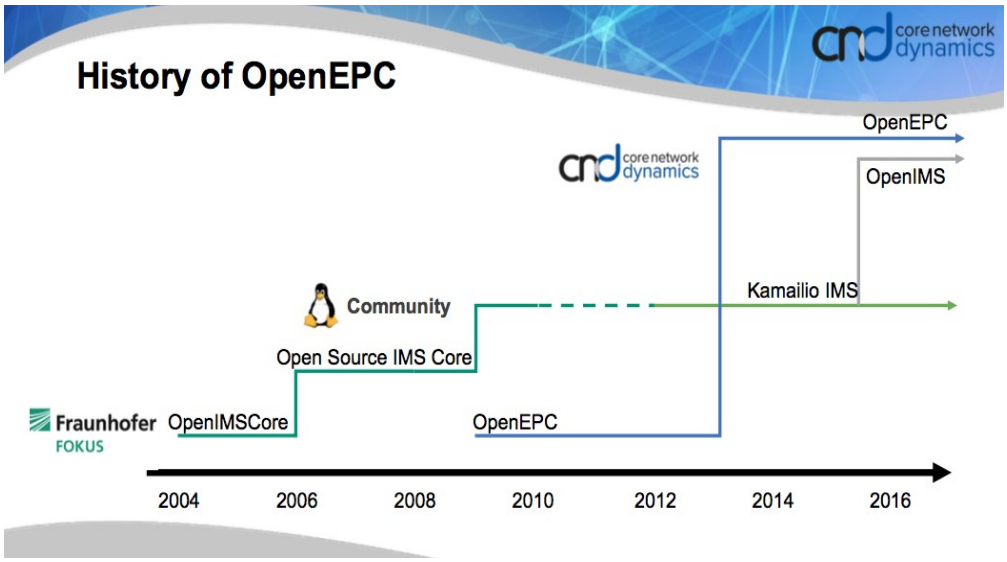


Figure 12. Historical Overview on OpenEPC.[12]

3.5.2. Brief Architectural Overview

The OpenEPC owing to its high complexity is provided in a typical setup having all the required configurations which could be used out of box. OpenEPC is delivered in a pre-configured and pre-provisioned system which could be used as virtual machines in a variety of hypervisor platforms such as VMWare Workstation, VMWare vSphere ESXi, Dockers etc. All the functional elements could be instantiated in single system or multiple instances can be merged into a single instance e.g. the Serving and PDN gateway is collocated into SPGW as shown in Figure 13 below in order to have less network latency. On the other hand, all functional elements can be instantiated multiple times also for load balancing etc.[12]

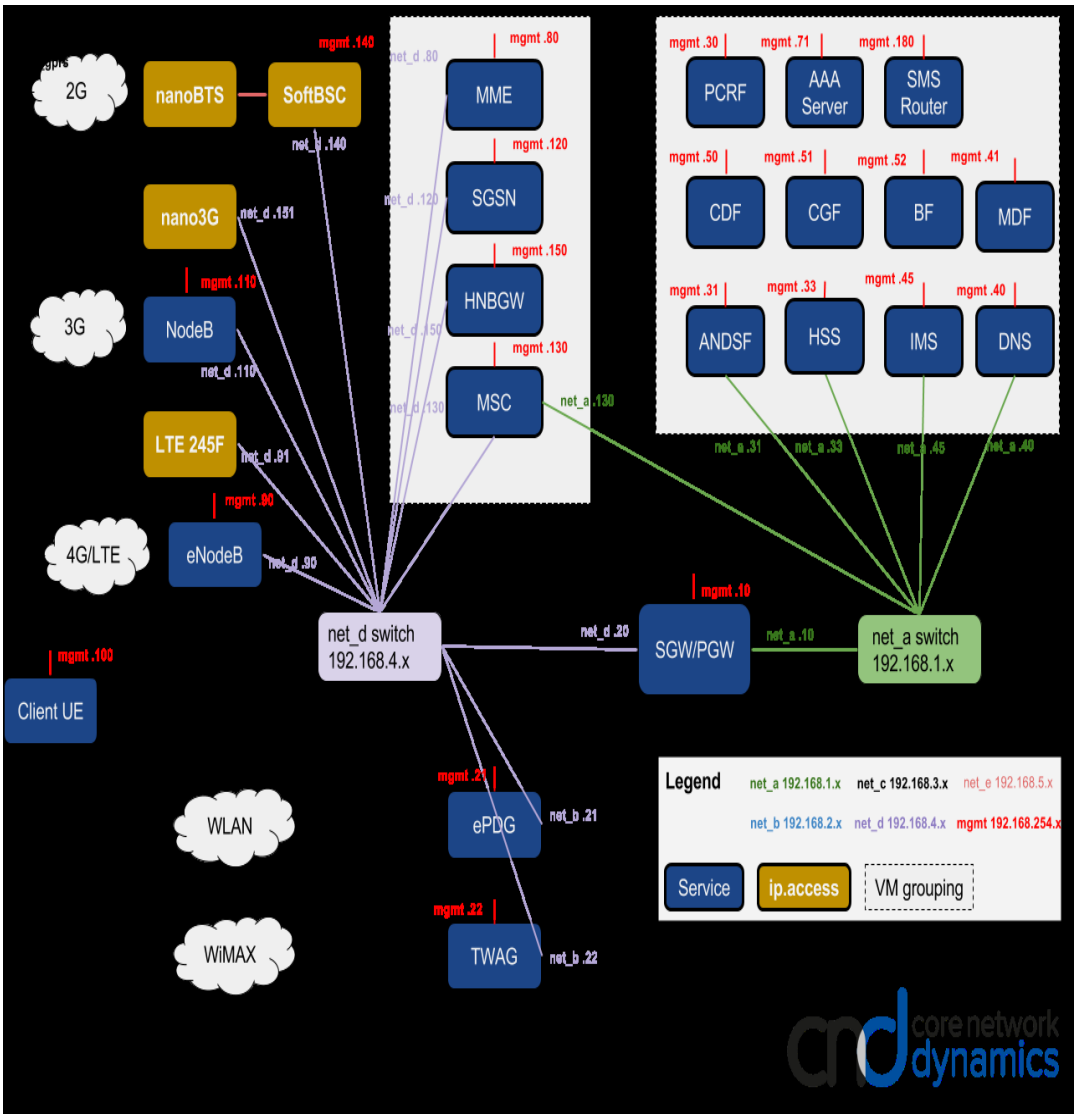


Figure 13. OpenEPC typical setup.[12]

As far as the network is concerned, there are several LAN segments which could be used. Figure 13 illustrates all the LAN segments with their respective IP pools inside the Legend. We will observe here all the LAN segments individually[12]:

mgmt Each component in OpenEPC has an mgmt interface i.e.192.168.254.X for interconnection and to have individual access to each component. It serves as a management network which is responsible for signaling such that most of the signalling i.e. Diameter goes through the mgmt network. The DNS within OpenEPC also uses mgmt interface at 192.168.254.40 as nameserver. The default gateway for NAT and DNS provided by VMWare is on 192.168.254.2.[12]

net_a It is the actual PDN providing access to the internet which connects the PGW and the application functions. It also serves as the default gateway for IP routing on the data plane path.[12]

net_b This segment is referred as the EPC backhaul which connects access network gateways i.e. ANGw, ePDG to the packet data network gateway. [12]

net_c It is the network segment exposed directly to the UE either through access network gateways or through eNodeB/nodeB emulations. IP address allocation with maximum mask for UEs is done on net_c in the PGW.[12]

net_d Connecting the RAN to the EPC core network, net_d serves as a 3GPP radio access backhaul. Any 4G physical device trying to build an S1 connection with the MME should be bridged to this network.[12]

net_e It is the 2G RAN segment which connects BTS to the BSC.[12]

net_f 3G RAN segment which connects the HomeNodeB to 3G access concentrator.[12]

an_lte, an_umts, an_wifi, an_wimax The eNodeB emulation uses these virtual interfaces to connect emulated client to EPC according to the LTE or legacy networks.[12]

3.5.3. Basic Installation

The basic installation of OpenEPC inside a virtual machine is performed by following steps mentioned below:

Step1: Install Ubuntu

Install Ubuntu and make auto login root i.e. root privileges. Perform a full system upgrade.

Step 2: Initial Preparation

Install subversion to check out OpenEPC setup and scripts from the source code repository.

Step 3: Network Topology Creation

Using 'make_udev.sh' at /opt/OpenEPC/sbin/net/, rename all the interfaces from eth0/eth1/wlan1/wlan2 to net_a/net_b/net_c/mgmt etc. according to network topology depicted in Figure 13. It is important to note that every interface should be matching the corresponding MAC addresses of individual vLAN.

Step 4: Installation

Finally run the 'install_system.sh' script to compile all source code and to install required tools, libraries, default databases etc. If you are not doing installation for the first time, remember to remove the 'install_config.sh' file before using the install script.

Step 5: Execution

Then execute individual services by using operations such as start hss, stop hss etc.

3.5.4. Components Allocation within OpenEPC

The component allocation within individual VMs for LTE-only access is listed in Table 2. The minimal setup that would allow to accomplish the entire configuration requires 7-8 VMs but here only VMs requisite for LTE access is described. The components represent modules which either runs in screen or at back end providing the requisite functionality and support. It is important to note that the EPC-Enablers will act as a DNS and gateway to other systems[12].

Table 2. Modules allocation within VMs[12]

VMs	Component Allocation
EPC-Enablers	HSS, PCRF, CDF, CGF, BF, ANDSF, Web GUI, DNS, IMS, NAT, AFs, HTTP-Proxy, MDF
SPGW	LMA, PCEF
eNodeB	eNodeB emulation with Wifi/Ethernet
Client	Client Mobility Manager
MME	MAG, BBERF

3.5.5. Components within OpenEPC

This section will give a detailed overview on all the components within OpenEPC.

3.5.5.1. Mobility Management Entity

The MME having control plane functionality to ensure authentication and authorization is implemented in OpenEPC based on the 3GPP TS 23.401 standard[12]. Like any proprietary hardware, MME, connects to the eNodeB on S1AP interface, with SGW on S11 interface and provides connectivity establishment along with mobility management for the UEs. [12]The MME in OpenEPC has all the prerequisite modules and protocol stacks. As depicted in Figure 14, 'mme' being the central module is supported by GTP, nas, client_s6ad, and S1AP protocol stacks as mandatory protocol stack modules. Since the MME is only part of the control plane so the GTP module has only GTP-C functionality. This module exports functions to retrieve or add information element to message structures which consists of list of information elements in a general message structure. It can update the location of UE through diameter protocol over s6a with the HSS which is located in EPC-Enablers as a node which runs in a screen at back end.[12]

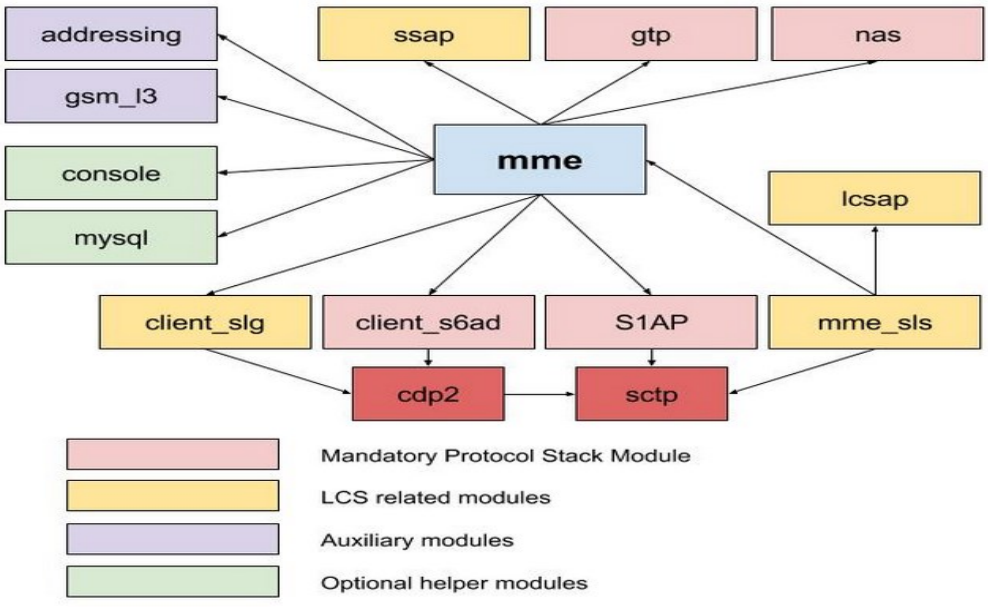


Figure 14. MME in OpenEPC.[12]

The S1AP stack is used for network related information from the eNodeB. The main ‘mme’ module stores the state of the attached UEs internally in a hash table of configurable size. The SCTP module implements SCTP socket functionality. Having Diameter based S6a interface, the client_S6ad module exports functions to handle S6a messages. The ‘nas’ module with its encoding and decoding functionality for NAS messages, exports functions i.e. create, encode, decode etc. related to NAS messages. The ‘addressing’ module having the capability of retrieving addresses using DNS and SNAPTR lookups determines the respective SGW the client will attach to. The ‘mme’ module make use of the MySQL module to export list of console commands in order to view the internal state of the mme.[12]

3.5.5.2. Collocated Serving and Packet Data Network Gateway

To reduce signaling latency on similar GTP interfaces SGW and PGW are replaced with a single entity in OpenEPC by disabling the SGW completely and configuring the PGW as SPGW. So, the SGW and PGW can run as a single component in OpenEPC for LTE/EUTRAN access only. Such a deployment is achieved by implementing S11 interface functionality in the local mobility anchor (LMA). It is important to note that the PDN connectivity establishment on gateway could be done on SPGW over both GTP and Proxy Mobile IP. In order for the SPGW to have accurate functionality gtp_s5s8, gtp_s11, GTP and routing _gtpu modules are requisite. Figure 15 gives an account of all the modules present with SPGW and the protocol stacks associated with it. ‘Lma’ module in SPGW is concerned primarily for the functionality of Control plane. It can maintain its own state machine for session management as the LMA module supports S5/S8 and S4/S11 using gtp_s5s8 and gtp_s11 modules respectively. The routing modules as depicted in Figure 15, takes care of the Data plane functionality of OpenEPC core network. Gw_bindings, being the central module in OpenEPC has all the information of the subscribers related to Gateway binding, access point names, bearer information, Policy charging and Control information.[12]

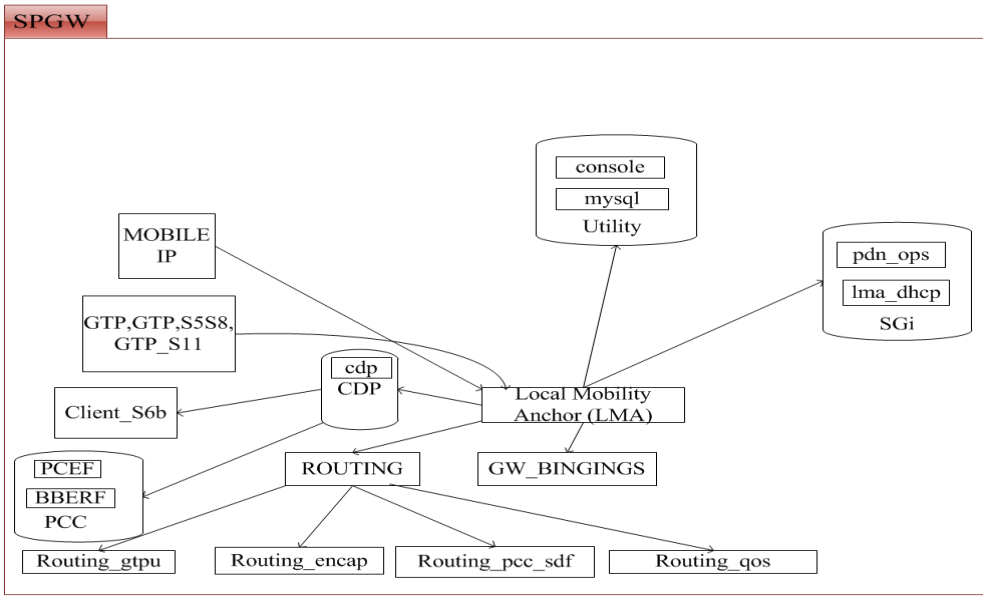


Figure 15. Collocated SPGW architecture in OpenEPC.[12]

The ‘client_S6b’ module implements the ‘S6b’ interface with the AAA server and uses ‘cdp’ module for Diameter communication. The ‘pdn_ops’ module uses a database to manage IP address blocks and together with the ‘lma_dhcp’, it forms the ‘Sgi’ interface. IP allocation is done either through radius or by SPGW using ‘lma_dhcp’ and ‘pdn_ops’ modules. Since the SPGW is capable of IP address allocation so upon attachment the ‘lma_dhcp’ will perform a DHCP request on behalf of the UE on the ‘Sgi’ interface. The then retrieved IP address will be allocated to the UE. This module also notifies the rest of the PDN network about a particular node where packets for UE has to be sent by having ARP communication. Finally, the ‘pcef’ module has the necessary functionality to handle the messages for attachment and PCC rule management for a UE.[12]

3.5.5.3. EPC-Enablers

EPC-Enablers serves as a node within OpenEPC having all the components shown in Figure 16 running as screen in standard Ubuntu virtual machine. All the IMS and other application functions is implemented here. The network address translation is performed here using the ‘DNS’ module present at mgmt.40 which serves as a primary DNS nameserver for all the components within OpenEPC. Moreover, the authentication of all the subscriptions are performed in the EPC-Enablers, in the HSS for 3GPP and AAA server for non-3GPP access.[12]

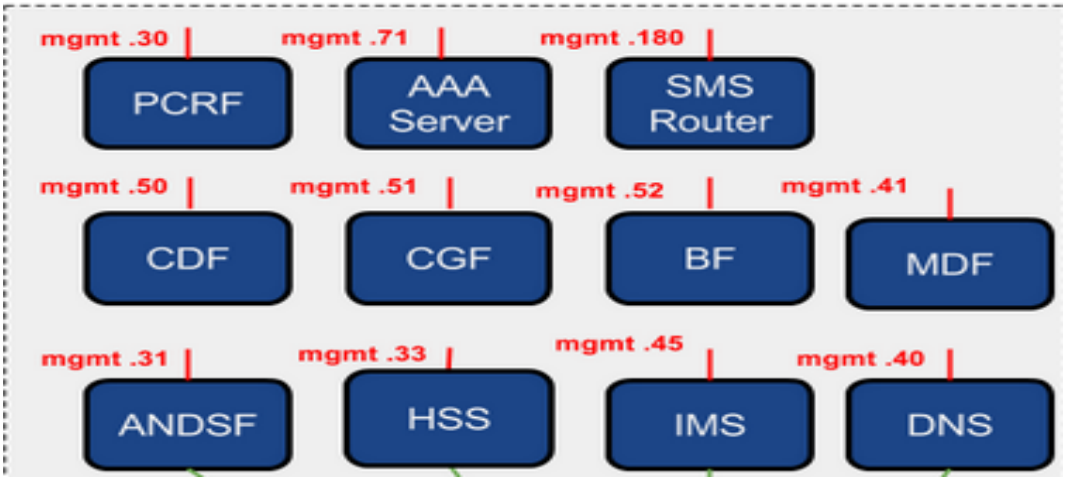


Figure 16. EPC-Enablers.[12]

3.5.6. Subscriptions and Security

This section will go through the essential details of a subscribers provisioning and authentication within OpenEPC. The credentials and identities which are requisite for connection establishment of a UE is mentioned in this section.

3.5.6.1. Identities within OpenEPC

OpenEPC uses several identities as part of subscription of a user. Those identities are mentioned in this section. Public Land Mobile Network identifier (PLMN ID) uniquely identifies a mobile network. PLMN ID consists of MCC and MNC which are mobile country code and mobile network code respectively. The PLMN ID is country specific but the OpenEPC in the default configuration uses the universally test PLMN Values i.e.01 as MNC and 001 as MCC values. Yet the MME in OpenEPC could also be configured with the network specific PLMN values which in case of 5GTN is 447 and 27 respectively. It is important to note that with the current settings in OpenEPC, 5GTN SIM cards of provisioned with the test PLMN values will be attached as roaming users. Then the International Mobile Subscriber Identity (IMSI) allocated to each mobile subscriber consists of 15 digits. It consists of MCC, MNC and MSIN (Mobile subscriber identification Number) e.g. 001011234567890 is the IMSI for Alice user provisioned within OpenEPC. The MNC and MSIN together is termed as NMSI (National Mobile Subscriber Identity). UEs are also allocated temporary mobile subscriber identity i.e. TMSI which is valid only within the area controlled by the MME. It changes any time while moving to different geographical locations and is used mostly for paging procedures. Moreover, if different MMEs are grouped for load balancing etc., unique identification of MME is performed by GUMMEI (Globally Unique MME identifier). The GUMMEI is formed of MCC, MNC, MME Group ID and MME code altogether so that an MME identifies another MME from an MME group by an MME code. For location tracking RAI (routing area identity) and TAI (tracking area identity) is used. The TAI has TAC (tracking area code for location update which in case of 5GTN is 137. [12]

An Access Point Name (APN) is requisite for a connection establishment which represents the name of GGSN/PGW. APN is composed of network identifier and Operator Identifier i.e. APN NI and APN OI respectively. APN NI defines to which external network the PGW/GGSN is connected to and APN OI defines the PLMN in which the PGW/GGSN is located. OpenEPC uses ‘default’ as APN NI and apn.epc.mnc001.mcc001.3gppnetwork.org as APN OI. As far as the domain names is concerned, OpenEPC uses fully qualified domain names in every component and functionalities, mmec12.mmegi0123.mme.epc.mnc001.mcc001.3gppnetwork.org for mme and TAI uses tac-lb23.mmegi0123.mme.epc.mnc001.mcc001.3gppnetwork.org.

3.5.6.2. Authentication

Home Subscriber Server (HSS), referred to as ‘CHeSS’, being the central component is an evolution of the HLR, holds the subscriber profile information. It uses Diameter as a communication protocol. The HSS has a common database which is used on multiple communication layers i.e. UTRAN/E-UTRAN on the ‘S6d’/’S6a’ interface, ANDSF & PCRF on the ‘Sp’ interface, IMS applications on ‘Sh’ interface and IMS functions on ‘Cx’ interface. HSS serves as a generic database like in case of IMS, application services can store generic data attached to the subscriber profiles. It is quite secured in a way that it only communicates with known and authorized nodes. On the operational level, operations are restricted based on peer by peer basis. Figure 17 gives an account of the modules functional within the HSS and the interconnection via different interfaces.[12]

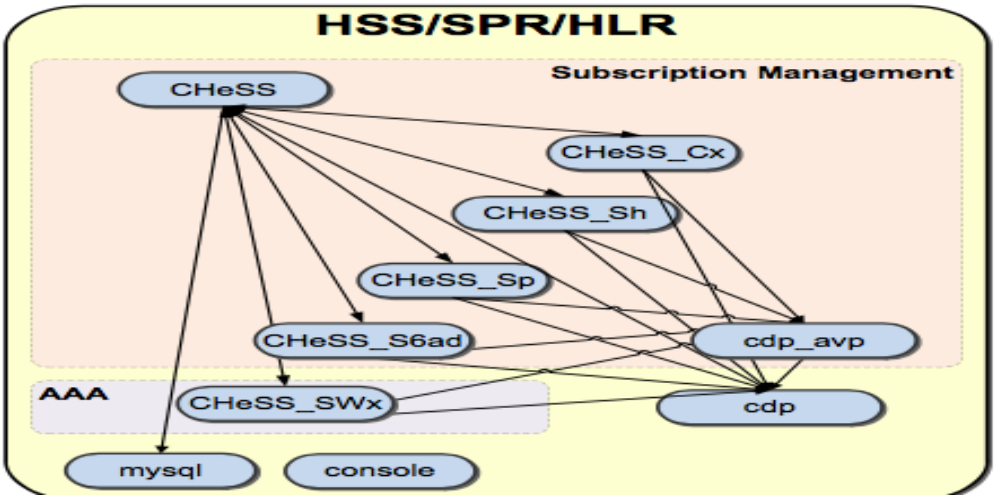


Figure 17. HSS in OpenEPC release 7.[12]

3.5.7. Policy Control and Charging

The PCRF, PCEF and BBERFs together forms the Policy Control and Charging (PCC) functionality in OpenEPC. Diameter interfaces are used for all sort of communications between components using different reference points. Rx, Gx, Gxx, Sp, S6a are the interfaces part of the PCC communication. Applications Functions (AFs) uses 'Rx' interface through PCRF to implement PPC rules during the EPC connectivity. The 'Gxx' interface as illustrated in Figure 18 like 'GX' is used between the PCRF and the BBERF to manage IPCAN sessions for QoS enforcement and gating control. But contrary to 'GX' while using the 'Gxx', during handover procedures, multiple BBERF sessions are maintained in parallel and rules are being migrated between them. The 'Sp' interface used between the PCRF and the HSS to retrieve data related to the user profile.[12]

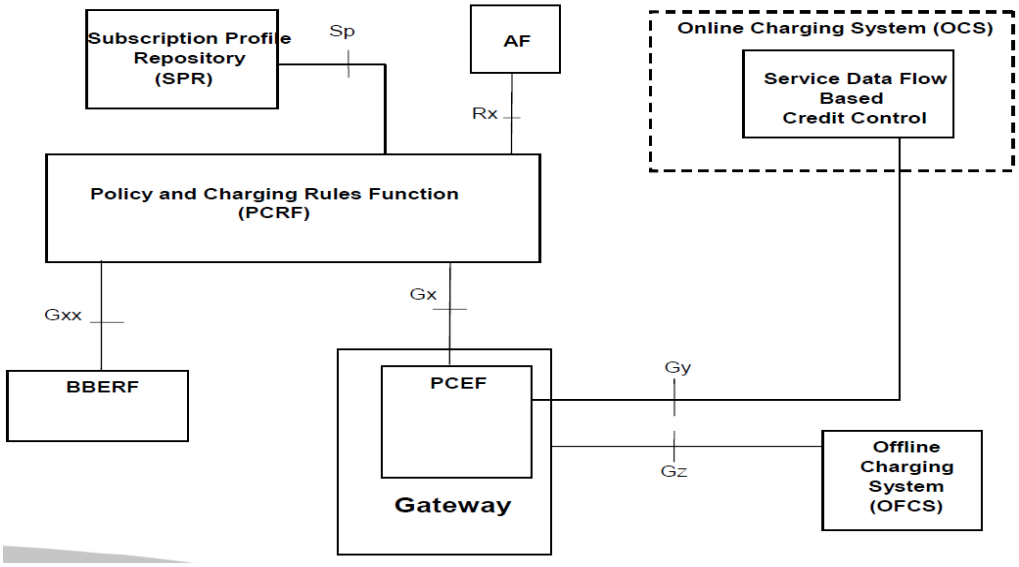


Figure 18. PCC architecture.[12]

Having specified session parameters 'AF' describes QoS and charging rules to be applied for the UE traffic on the network segments between the client device and the PDN gateway. Implementing PCC rules, IPCAN sessions is managed to have charging and gating control by using 'GX' interface between PCRF and PCEF. PCEF maintains a Gateway Control Session (GCS) with PCRF for each subscriber attached to the network. PCRF also creates default bearer QoS based on the HSS.[12]

3.5.7.1. PCC Bearers Binding

Bearer is an information transmission path having defined capacity, path and bit error rate. A bearer can be default or dedicated, where a default bearer is allocated with each PDN connection providing basic connectivity and a dedicated bearer is assigned to have a specific level of QoS and charging. QCI, ARP, MBR, GBR are some of the attributes of a dedicate bearer. PCC functionalities are executed at the course of a bearer level. An EPC bearer can identify packets between the UE and the PDN gateway which have common QoS. It usually deals with scheduling, queue management and rate shaping policies. Bearers encapsulates data packet flows such that a packet flow contains the source and destination IP address with a defined port and protocol ID.[12]

3.5.7.2. PCC Operations

The Policy and Charging Control functionality is an imminent factor from an operator's perspective as the QoS and Charging depends on it. OpenEPC, like a traditional LTE network has a PCC functionality managed by the PCRF module where a UE, trying to establish a connection, is dealt based on the subscriber profile already present in the HSS repository.[12] The whole process consists of 15 steps in total which can be seen in Figure 19 below.

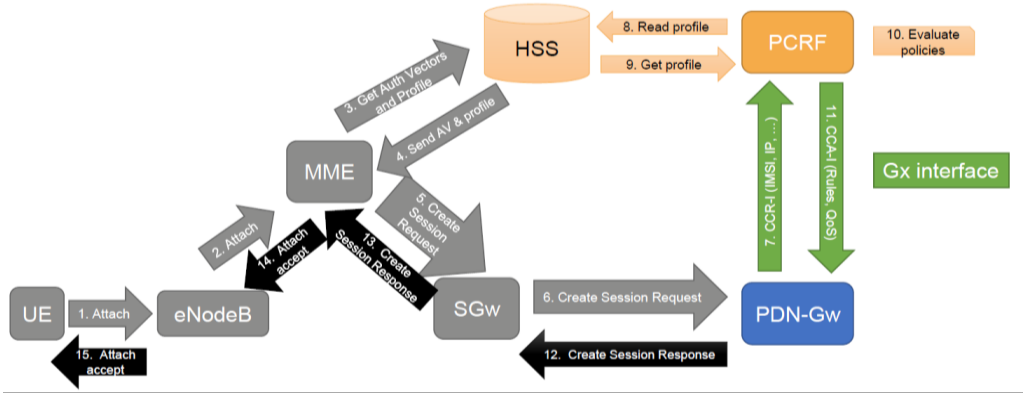


Figure 19. IPCAN Session Establishment.[12]

3.5.8. Web GUI and Provisioning

OpenEPC offers a provisioning interface i.e. Web GUI at 192.168.1.32. The Web GUI is running on top of an Apache 2 platform where all the provisioning within OpenEPC is performed. For example, the HSS provisioning, which is split into two parts i.e. HSS provisioning from IMS and SPR provisioning from EPC, allows to provision all information regarding subscribers for entire EPC and IMS setup.[12]

4. OPENEPC IMPLEMENTATION AND 5GTN

Contrary to the conventional end to end proprietary 3GPP EPC, the growing need for a virtual core network having open interfaces with management and orchestration support, OpenEPC as a virtual LTE core network could be deployed as virtual network functions in a server virtualization platform. Keeping in view the demand for different IoT use cases along with the growing need for an NFV based EPC, this chapter will go through every bit of details to explain the implementation of OpenEPC in a non-virtualized network i.e. 5GTN. It gives an insight to the journey of making a testbed on 5G PoC which could be utilized for ample of use cases.

4.1. Introduction

Having an existing LTE network at CWC, University of Oulu, owned by 5GTN research group which is open primarily to be used for research purposes, provides a possible opportunity for service providers and researchers to come up with innovative use cases. Thus, providing the testbed platform for researches leading to future cellular wireless infrastructure. The combination of mobile networks and technologies from IT is taking cellular networks to new horizons. The concept of virtualization is a cutting edge of technology which when implemented to mobile networks leads to immense innovative possibilities. Virtualization of the LTE core network elements to be used from a cloud is to avoid conventional off the shelf proprietary hardware and provides a virtualization platform capable to be software defined to orchestrate the network. Once virtualized the network then can be further modified accordingly with the growing needs. For Example, in order to have support for SDN functionality with a Centralized Controller and Network Programmability Support, the network elements should be virtualized as NFV serves SDN by implementing network functions in a software manner [2]. The SDN controller then would be running on a cloud which could be migrated to different locations according to network needs [2]. 5GTN keeping in view the growing demand for NFV and SDN has laid a foundation stone towards Network Functions Virtualization by implementing OpenEPC, a virtualized LTE core network within the LTE based test network, is a set of cloud native applications. Hardware independency and the use of standard Linux platform makes the system compatible with NFV[12 1]. The dynamic configuration of the elements is possible using REST APIs and JSON-RPCs protocols[12] and above all OpenEPC has a support for ETSI MANO components as it could be integrated with Open Baton, NFVO, EMS, VNFM etc. Though the IT servers at University of Oulu do not have yet support for Open Stack environment on which in future most of the SDN and NFV deployment will be based on but they do have other alternative virtualization server environments e.g. VMWare vSphere on which OpenEPC is currently up and running with each major core network elements as virtual instances. This chapter will go through the details of implementing a virtual core network i.e. OpenEPC on 5GTN infrastructure.

4.2. Architectural Overview

Since 5GTN has always been evolving towards latest wireless cellular technologies from its inception, there were different stages before the network got into existing shape. Different phases of 5GTN i.e. before and after the implementation of OpenEPC is hereby projected in this section to better understand the evolution scenario of 5GTN architecture. 5GTN's RAN is using Nokia's Evolved Packet Core over VPN from the SRN (shared reference network) which exists in Tampere, Finland as shown in Figure 20. Though there are several bottlenecks in such a setup i.e. the bandwidth limitation on the Juniper switch which in this case is 300MHz at SRX240 Juniper switch and even less at the VPN switch used by Nokia at the moment which is shared equally among the total number of connected users having more network latency yet it was deployed initially in 5GTN and can be seen in Figure 20 in details.

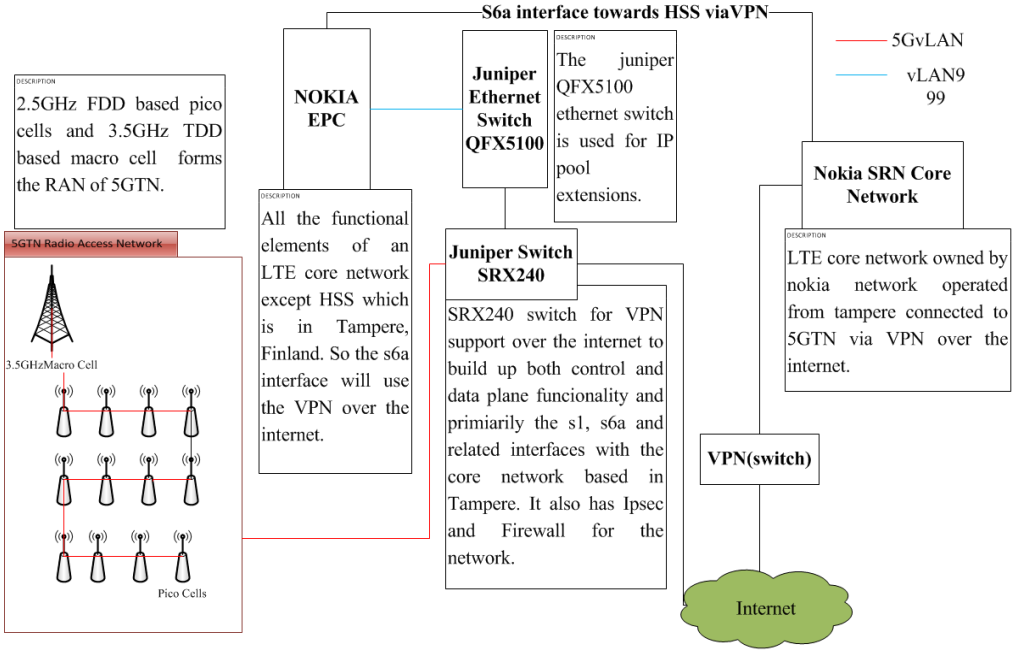


Figure 20. 5GTN having SRN connection from Nokia core in Tampere, Finland.

The Juniper switch provides two IP pools, of which one is the IP pool named as 5GvLAN assigned for the 5GTN radios i.e. the 2.6GHz Pico cells and the 3.5GHz Macro cell to be connected to. While the other IP pool is for communication with the other Juniper switch which then provides an IP pool called as vLAN 999 reserved for the Nokia Evolved Packet core which is under installation and deployment in University of Oulu, Linnanmaa campus premises. In case of the Nokia EPC, the connection with HSS for authentication over s6a interface will still be in Tampere and could be reached by VPN over the internet. 5GTN doesn't have any access to the core network components so none of the Control and Data plane traffic could be monitored. Moreover, network performance optimization which is inevitable is not possible with this sort of setup deployment. Network function virtualization and Software Defined Networking, the future of existing networks which is expected to be deployed in 5G, created on demand for a virtual core network to be deployed in 5GTN. OpenEPC by CND having a virtualization prototype implementation capability of the 3GPP EPC[1]

is introduced to the test network primarily to have a foundation for a virtualization platform testbed and also to get rid of proprietary hardware. Figure 21 shows an extension to 5GTN with OpenEPC being introduced as a core network on the same virtual LAN to which the RAN is connected to. Contrary to the conventional EPC, OpenEPC has each functional entity running as virtual machines on top of a commercial hypervisor, VMWare in this case. Every virtual machine has several virtual network cards from 5GvLAN keeping in view the network topology defined in OpenEPC as shown in Figure 13 for each component. 5GTN currently owns an LTE license of OpenEPC limited to LTE (data only) which will be further enhanced in future keeping in view the use cases.

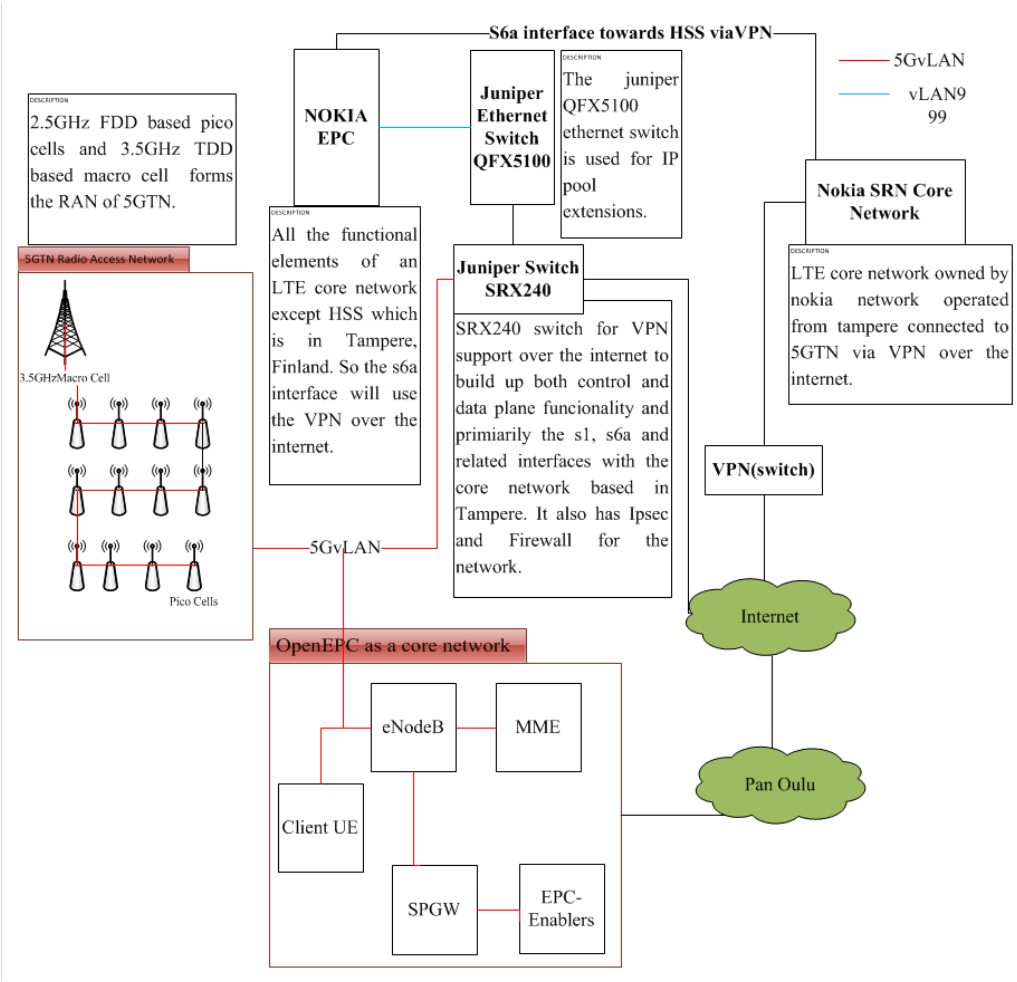


Figure 21. 5GTN after OpenEPC implementation.

4.3. OpenEPC Integration within 5GTN

As mentioned earlier in the Introduction chapter, Nokia small cells i.e. eNodeB with LTE band 7 and 42 licenses forms the RAN of 5GTN while the SRN based Nokia Network's Evolved Packet Core forms the core network part. The core network is reached from 5GTN by a VPN connection over the internet. Since the VPN box used for this purpose has bandwidth limitations so it appears to be a bottleneck for an efficient testbed infrastructure. With extensive researches on 5G from radio to network

and applications perspective, Network Function virtualization seems to fulfil the growing needs of a major network evolution from existing networks [2]. To have the capability of NFV in a network it should have a supporting virtualization infrastructure so 5GTN at CWC with the collaboration of University of Oulu's IT department reserved a locally administered cloud. VMWare ESXi virtualization platform in place at the University servers which will be maintained by the IT department having the LTE core network in place as virtual instances. OpenEPC is the virtual core network integrated with 5GTN on the same virtual LAN it has been using for the SRN connection. The small cells within the network are then commissioned with respect to different core networks so that it forms an 'S1' connection with the MME and build up respective Control and Data path. A deep analysis of the network from each node will make the deployment scenario easy to understand.

4.3.1. OpenEPC on 5GvLAN

5GvLAN has a default gateway at the juniper SRX240 switch which offers flexibility for delivering secure and reliable data over IP. Having a support for WAN, LAN and POE connectivity the SRX240 Service Gateway provides services like IP Security (IPsec), virtual private network (VPN), and firewall.[16] Then comes the Juniper QFX5100 Ethernet switch providing LAN extension to a different IP pool named as vLAN999 reserved for Nokia Air frame Evolved Packet Core. OpenEPC is installed on University virtual servers where each network element of the OpenEPC has several vLAN connections from 5GvLAN.

Initially the 'iso images' with .vmx extension provided by CND for the core network components i.e. client UE, eNodeB (emulated), MME, SPGW and EPC Enablers as preconfigured virtual machines are uploaded to the vSphere ESXi by removing the already present network topology. It is important to note here that the preconfigured virtual machines can be created from scratch as we have access to the repository created by CND to download and update the required binaries. Having uploaded the images for every individual virtual machine and adding required number of virtual network interfaces on the vSphere ESXi server keeping in view the CND's OpenEPC network topology diagram, the network topology is created by a predefined script i.e. root@OpenEPC:/opt/OpenEPC/sbin/net/#./make_udev.sh and then running the. /install_system.sh script to install all the prerequisite configurations. OpenEPC is setup in such a way that the virtual network interfaces from 5GvLAN are provided in each OpenEPC nodes along with a vLAN from a public network for resolving domain names out of the network on the internet which in our case is Pan Oulu. Since internet connectivity within each virtual machine is imminent for any updates and downloads so Pan Oulu serves the purpose. Though the OpenEPC has a NAT at 192.168.254.X to resolve all the DNS queries within the network which should be set up in each virtual machine in /etc/resolv.conf but we placed another NAT i.e. 'Internet' to resolve DNS queries outside of the OpenEPC network on the internet and defined it again in /etc/resolv.conf.

Now, we will look into network interfaces on individual virtual machines to better understand the way OpenEPC is configured. We have five virtual instances in total, of

[16] J. Networks, *SRX240 Service Gateways Hardware Guide*, 1st ed. California, USA: Juniper Networks. Inc., 2014, pp. 3-7

which eNodeB and client-Alice is an emulation of LTE RAN. They have no radio interfaces at all and the user traffic is directly encapsulated in GTP-U tunnels without any radio stack processing[12]. The client-Alice acts as a DHCP client for the emulated eNodeB having a mobility manager support to switch between different legacy, 3GPP and non-3GPP access networks[12]. So, the UE and eNodeB emulation implementation does not match any of the 3GPP UE categories as they describe UE's radio capabilities. eNodeB has three virtual LAN connections on 5GvLAN i.e. net_c, net_d, mgmt and one from Pan Oulu i.e. Internet. The details of interfaces for each virtual machine can be seen in Figure 22 below. Figure 22 provides details about the virtual network connections on individual nodes within OpenEPC on the University servers. Since 5GvLAN is configured in such a way that every node on it has to have a static IP assigned manually so the static IPs assigned to each virtual machine by default can be utilized by making the default topology using /opt/OpenEPC/sbin/net/# ./make_udev.sh in every virtual machine. 'net_a' is the actual PDN which connects the PGW and the application functions, 'net_d' is the 3GPP radio access backhaul, 'net_c' is for the network segments exposed directly or through emulations, 'an_lte' is a virtual Ethernet segment used by the eNodeB emulation and 'net_b' is the EPC backhaul which connects access network gateways[12].

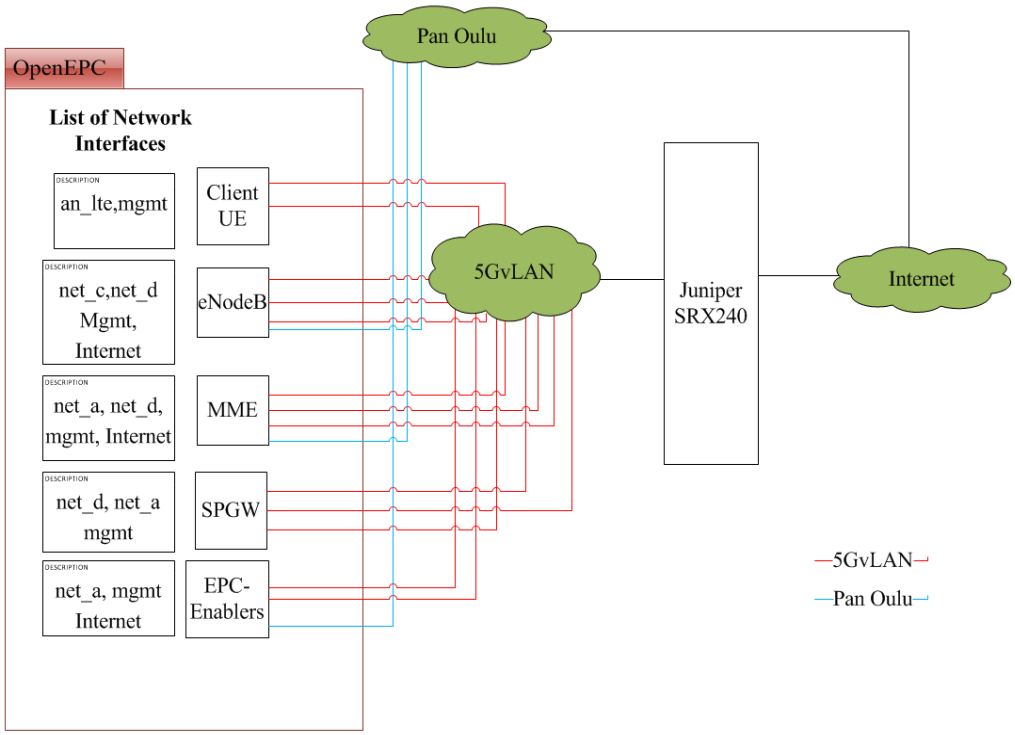


Figure 22. OpenEPC on 5GvLAN.

4.3.2. Bridging the 2.6GHz Pico Cells with OpenEPC

The eNodeB emulation can be used in parallel with a physical eNodeB but can also be shut down while having a real eNodeB connected to OpenEPC[12]. Bridging is needed to deploy such a setup, a terminology used within VMWare when a virtual network interface has to be connected directly to the physical network, where the Pico cell in

our case is connected to the 'net_d' interface of OpenEPC. Since the 'net_d' represents S1 interface in the standard setup so we need to reconfigure the VMWare to use a bridged interface on the MME VM, instead of 'net_d' LAN segment. As for the IP pool is concerned, we need to assign IP address to the external eNodeB which should be in the same network as the emulated eNodeB uses i.e. net_d. If any other IP than 'net_d' is used appropriate routes in the SPGW and MME has to be added[12].

While configuring a physical eNodeB, a DNS entry for the Tracking Area Code to which the eNodeB is configured needs to be added which is used for SPGW selection during attachment procedures. This is done in the Web GUI by navigating to Tools> PDNS>Records and searching for 'tac' records. The OpenEPC uses **tac-lb01.tac-hb00.tac.epc.mnc001.mcc001.3gppnetwork.org** for its default eNodeB. So, the TAC for the physical cell which is 137 in our case needs to be converted to hexadecimal and placed in lower & higher bits in the OpenEPC 'tac' format. It is important to consider that if there are less than two significant digits in TAC, '0' should be inserted at the left side.[12]

Since commissioning of the Nokia's Pico cell to match the configurations of MME is equally important to use it with OpenEPC so using BTS site some major modifications are made. The Control and Data plane IP is set to the IP of the OpenEPC MME to have an 'S1' functionality. The 'MNC' and 'MCC' i.e. PLMN is changed to 001 and 01 respectively as it is universally used in test networks. Last but not least, the 'TAC' is also changed within the eNodeB itself. 5GTN remote RAN with the SRN from Tampere core uses Top Servers for synchronization which in case of using with OpenEPC is changed with GPS antennas. Having made the changes mentioned above, any cell connected on 5GvLAN should setup an S1 connection with the vMME.

4.3.3. Provisioning 5GTN SIM Cards

OpenEPC being implemented in 5GTN can be used out of the box for provisioning any SIM cards. OpenEPC is configured by default to use CND SIM cards as home users i.e. the MME has MNC001 and MCC01 which are test PLMN values being used universally. 5G SIM cards when provisioned to be used with OpenEPC are registered in the network as roaming users as the base station and MME both has the CND's default configurations. This can be overcome by changing configurations within MME or using another MME with 5GTN's PLMN values, so that we can commission the base station with the same values. The provisioning can be done initially by introducing the user's data to the HSS data base in the root@EPC-Enablers:/opt/OpenEPC/sbin/provision#by using several scripts e.g. ./hss_add_user.sh and it will ask for some parameters like OPc, IMSI and Secret key. The same configurations could also be performed in Web GUI which is accessible from either the SPGW or EPC-Enablers machine. In the Web GUI, running on top of an Apache2 platform[12] we must navigate to the HSS/AAA option and add all the required subscription related data for an IMSI. The actual provisioning of a SIM card can either be done in the GUI or via HSS console in EPC-Enablers VM using 'usim. provision' command[12].

4.3.4. Accessing Campus Free LTE

5GTN proudly presents a free LTE service running on multiple core networks. A user while connecting to a cell configured to be connected to OpenEPC will have an LTE access by the title of 001. Assuming that the user already provisioned in the HSS, can connect to LTE only if proper APN is configured in the user equipment. With the current setup, 'default' APN is used but can be changed to 5GTN's APN by defining it in the Web GUI. There is an APN configuration profile in AAA>EPC SPR where we can add new or modify the existing APN and it should than also be assigned to the user's HSS profile from AAA> EPC SPR> Subscriber Identities>Allowed APN Mappings[12]. Moreover, the new APN should also be included in the DNS data base and also to the SPGW[12].

4.4. 5GTN Control and Data Plane Functionality with OpenEPC

To better understand the packet routes i.e. both Control and Data plane traffic for a UE connected to campus free LTE, a detailed analysis of the individual nodes is requisite. OpenEPC uses 'modules' terminology for components running at back end in each virtual machine. Most of the signaling goes through the 'mgmt' network i.e. 192.168.254.x using diameter protocol. The 'mgmt' network is used as a management network for an access to individual virtual machines. So, for DNS the /etc/resolv.conf should have the following contents:

- domain epc.mnc001.mcc001.3gppnetwork.org
- search epc.mnc001.mcc001.3gppnetwork.org
- nameserver 192.168.254.40
- nameserver 10.20.110.4
- nameserver 10.20.110.5

The IP address allocation for the UE in the access networks should be performed with maximum mask such that all the traffic between UEs goes through the ANGw and PGW, where QoS and Charging is performed.[12] All the machines are configured with the default gateway so as to have internet for updates.

The GTP module implements GTP stack for both Control and Data plane messages. Having the logic for GTPv2-C messages, the gtp module exports functions to add information to the message structures. It has the transport plane communication logic which consists of UDP port functionality and it can be configured to listen on multiple interfaces. For data plane traffic, the routing_gtpu module will create the necessary sockets.[12] The MME having Control plane functionality connects to the eNodeB and SPGW on S1ap and S11 interfaces respectively. The eNodeB and SPGW have the role of providing Data path for the UEs connected to the LTE access network while the MME has the role to ensure authentication, authorization as well as the connectivity establishment and mobility management for the UEs[12]. Since the PDN connectivity establishment can be done over both GTP and PMIP, the MME can be connected directly to the collocated S-PGW over S11 interface[12]. The MME being only part of the control plane has GTP-C functionality and can update location of the subscribers through Diameter protocol over S6a with HSS[12] as shown in Figure 23. The 'client_s6ad' module in the MME exports functions to create, send and handle s6a

messages[12]. The MME has several other modules which are discussed in details in chapter 3 of the thesis so here we will discuss only the main modules requisite for Control path.

The OpenEPC core gateways are collocated in a way that the SGW and PGW runs as a single component by implementing S11 interface functionality in LMA so that the signaling latency on similar GTP interfaces get reduced. The PDN connectivity establishment on gateway can be done on SPGW over GTP and PMIP both. The Local Mobility Anchor (LMA) and routing modules as depicted in Figure 23 realizes the core functionality for Control and Data plane respectively in case of SPGW. LMA maintaining its own state machine for session management supports gtp_s5s8 and gtp_s11 interface on GTP and PMIP. [12]

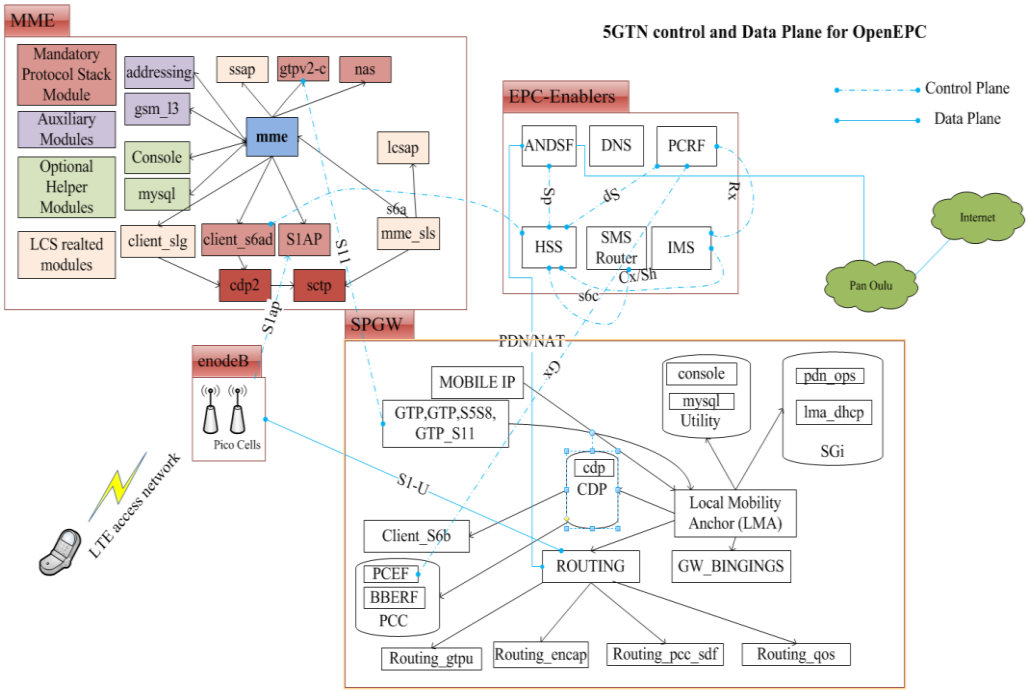


Figure 23. Control and Data Plane in 5GTN using OpenEPC.

A UE already provisioned within the HSS when connected to a Pico cell gets authenticated and attached having established an S1 setup with the MME using the S1AP module. The UE packets are transferred to the MME via NAS protocol encapsulated at the eNodeB within the S1AP[12]. The gtpv2-c in the MME being connected to the GTP over S11 interface is also a part of the Control plane as shown in Figure 23. IP allocation is done upon attachment by SPGW using the DHCP (lma_dhcp) module or PDN operation (pdn_ops) module by performing a DHCP request on behalf of the UE on the 'SGi' interface[12]. The module also takes care of the ARP to notify the rest of the PDN network of the node where packets for UEs need to be sent[12]. The Data path primarily consists of the S1-U connectivity between the eNodeB and the routing module within the SPGW having the support for GTP tunneling. Then the IP packets are routed using the default packet route i.e. 192.168.1.x set in the OpenEPC as shown in Figure 24 to the ANDSF in EPC-Enablers machine using NAT or PDN connectivity. From there using the DNS settings it goes to the destination address over the internet using the Pan Oulu gateway. In case of the

downlink traffic, the packets are routed all the way to the eNodeB and then to UE at 192.168.3.x using the same default route at 192.168.1.x between the EPC-Enablers machine and the SPGW.

```
root@spgw:~# ip r
default via 192.168.1.70 dev net_a
192.168.1.0/24 dev net_a  proto kernel  scope link  src 192.168.1.10
192.168.4.0/24 dev net_d  proto kernel  scope link  src 192.168.4.10
192.168.254.0/24 dev mgmt  proto kernel  scope link  src 192.168.254.10
```

Figure 24. IP route for the Default gateway.

5. RESULTS AND FINDINGS

This chapter will go through a series of network performance measurements to test the feasibility of a virtualization platform for LTE's EPC. After going through all the measurements and their conclusions, the research questions mentioned in chapter 1 should have logical and empirical answers. The primary purpose of the measurements is to find out how badly the downside of virtualization effects OpenEPC and either it is enough efficient like a commercial hardware. To draw a logical conclusion out of performance tests, we should take a look at the host server hardware and guest machines configurations first.

5.1. Host Server and Hypervisor Configuration

OpenEPC being installed on VMWare vSphere ESXi virtualization platform where we can create and run virtual machines and appliances[4]. vSphere ESXi is a bare metal type 1 hypervisor which could be installed directly on a server hardware as an operating system. The server can then be remotely accessed from a vSphere client which in most cases is an internet browser running as an application on top of Windows/Linux operating systems. Since in a data center there could be multiple server hardware in a cluster operating on different hypervisors, running multiple applications so there must be a central server managing all those resources. In case of ESXi hosts connected in a network, vCenter Server is a service which performs roles of a central administrator as it manages resources of all hosts in a network cluster[4]. The remote communication link from the vSphere client to the virtual machines running on a host server is depicted in Figure 25. In order for the client to access the virtual machines remotely there should not be any firewalls blocking the connection in between as can be seen in Figure 25 below.

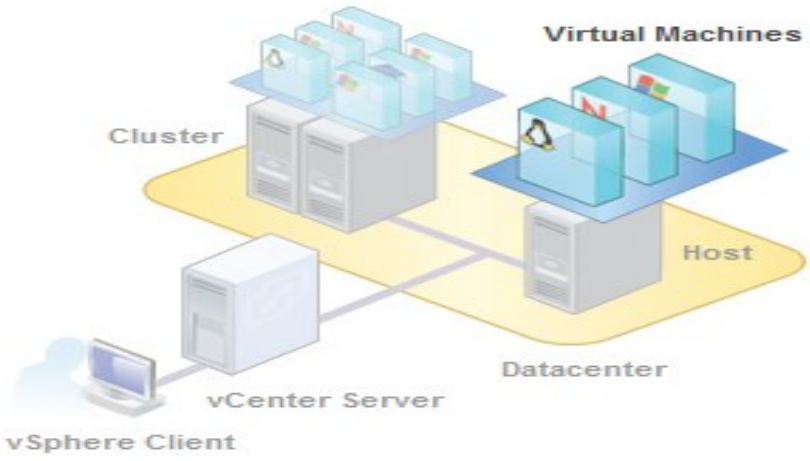


Figure 25. VMWare vSphere Server in a nutshell.[4]

5.1.1. Server Hardware

The Dell Inc. PowerEdge R820 server is being utilized as a host machine in the Oulu Data Center managed by the University of Oulu's IT department. Having four Intel Xeon E5 processors R820 is capable of handling heavy workloads. Since we expect a heavy explosion of data as the telecommunications network infrastructures is going to be virtual in the future mobile generations, the server hardware should thus be highly scalable. So, having 1TB memory support, R820 could be enough scalable as virtual servers running in a network cluster. The host machine holding OpenEPC has 16 internal hard drives with 32 cores in 4 core processor configurations with a 2.5MB cache per core holding up to 48 DIMMs. Capacity is much enhanced by having 16 internal hard drives and PCLe Gen3-enabled expansion slots. The Operating system for R820 could be VMWare vSphere Hypervisor or Citrix Xen Server, we used the former one in our case for OpenEPC. [17]

5.1.2. Guest Configuration

The resources allocated for guest machines which have the OpenEPC core network components as virtual machines are described below in Table 3 for individual VMs.

Table 3. Guest virtual machines' configuration on vSphere ESXi

VMs	Processors	Memory(MB)	Hard Disk
MME	2	2048	14.6484375
SPGW	4	4096	14.6484375
EPC-Enablers	2	2048	14.6484375
eNodeB	4	4096	14.6484375
Client User	2	2048	14.6484375

5.2. Load Testing

Load Testing is performed for both Control and Data plane traffic by registering a certain number of UEs initially and then, they send data continuously. Due to limited number of terminals, the interfaces are loaded by using an Open Source load generator. The tool generates CPU bound processes in the kernel also termed as worker threads which puts a specified load on the VM's resources. The idea is to check the stability of the testbed in different loading capacities. Along with the self-generated load and the normal processes running within each VM, actual UE's Data plane and Control panel traffic is also part of the load. We will try to come up with an approximate load, a single UE creates on both the Control and Data plane.

[17] Dell Power Edge R820 Specifications Sheet

5.2.1. Load Testing at the Collocated Gateway

The Serving and PDN Gateway are the two primary Data plane components in an LTE Evolved Packet Core which processes the actual user data traffic from and to the UE via GTP tunnels [13]. In OpenEPC these two components can be collocated as SPGW in a single VM which is done primarily by making some configuration changes at the PDN Gateway[12]. To provide better QoS to the end user, the SPGW should be optimized accordingly. MNOs should take into account an estimate of the number of users, the SPGW can hold with any existing resources so that upon requirement the resources could be enhanced to meet an expected number of users with no QoS compromise. After the deployment of OpenEPC in 5GTN, already explained in chapter 4, a performance evaluation is requisite to check the feasibility and scalability of the infrastructure. So, Load testing is performed at the SPGW by varying number of UEs. The terminals are generating constant data traffic which has very meagre amount of load on the CPU so loading the default gateway (for IP packet routing) with only actual users' data is not possible (as we are short of number of terminals). Since the load for few users is almost non-existent and performance estimation cannot be carried out without loading, so the vCPUs are loaded with an Open Source tool.

Imposing a scalable amount of load, 'Stress' is a work load generator[18] which creates actual CPU-bound processes to a system load[19]. The tool is scripted in 'C' and could be utilized for over 160 stress tests[20]. Stress-ng, an updated version of 'stress', was designed for numerous physical subsystems and kernel interfaces of different operating systems. The stress mechanisms grouped in classes has a wide range and are called stressors[18]. The mechanism we utilized is realized by the script: root@spgw: ~# stress-ng –vm *number of vCPU to be loaded* –vm-bytes 1G

- Where,
- --vm N = starts N workers spinning on anonymous mmap [20]
- --vm-bytes N = allocate N bytes per vm worker (default is 256MB)[20]

Before understanding the concept of load average, it is important to note that any running process inside a Linux environment is a process which is actually being executed in the processor at that moment. Since multiple processes keeps on executing and going out of execution on a single processor core by making use of a kernel specific version of the state machine. Whenever a process goes out of execution, the OS puts it in a holding state and it keeps on waiting for an I/O. Once the time slice of a CPU bound process expires, the Kernel pre-empt them. These processes are still ready to run and the kernel schedules other processes so as to make them executable again. Hence, the kernel calculates the running average after sampling data at some predefined time. It basically takes average of all the number of processes in the queue, ready to run but waiting for a time slice of the processor and the number of processes which are running currently.[19] Since, we have four cores in the SPGW VM, a CPU Load will be calculated on the scale of 4. Any load greater than 4 will compromise the QoS and smooth running of Users' data traffic because the kernel will put processes in a queue[19].

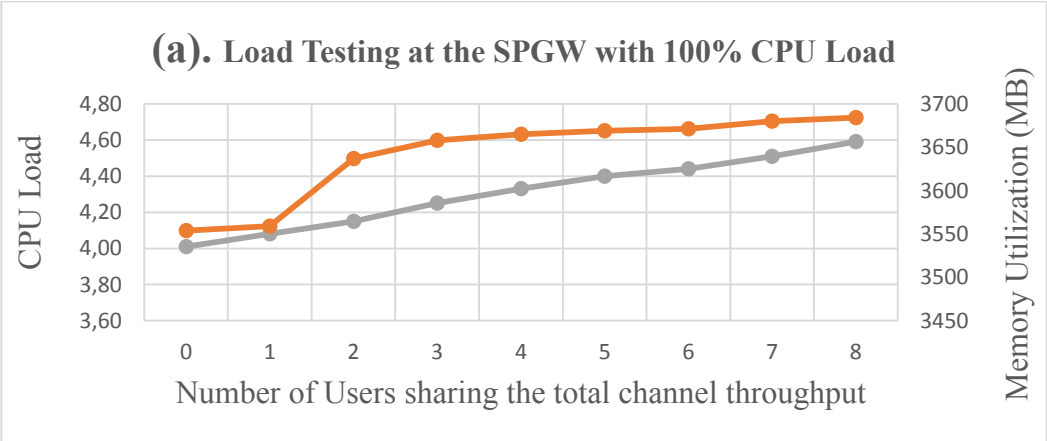
---

[18] Stress tool project page available at http://people.seas.harvard.edu/~apw/stress/.

[19] 'One Metric to rule them' by Federico Lucifredi available online at Admin Network & Security http://www.admin-magazine.com/Articles/Law-of-Averages-Load-Averaging.

[20] Stress-ng manual available at http://kernel.ubuntu.com/~cking/stress-ng/stress-ng.pdf.

Initially, the Average Load and the Memory Utilization for the SPGW machine is monitored by using ‘uptime’ script. The SPGW being idle gives a CPU load of 0 and a very less memory utilization. Then, the CPU is loaded 50% by running 2 CPU bound processes with a ‘vm’ data of 1GB each and no UE is connected up till now. The amount of %CPU load and %Memory utilization is depicted in Figure 27 whereas the actual values for both of them i.e. 2405 MB and 1.97 respectively can be seen from Figure 26. Now, in order to see the effect on both the CPU load and Memory utilization by actual UE traffic, a terminal is connected to OpenEPC via LTE radio i.e. 2.6GHz Pico cell. The terminal is then used to play a video streaming online on YouTube so as to generate data traffic flowing through GTP tunnels between the SPGW and eNodeB. As depicted in Figure 26 and Figure 27, the user traffic made a minor increment in both CPU load and Memory Utilization. It is important to note that the total channel bandwidth is equally distributed between the number of terminals being connected. The throughput on individual terminals when combine together will give the total throughput which in our case is around 20.90Mbps. Then, we kept on increasing terminals similarly to check the loading on SPGW. So, when the CPU is half loaded with the CPU bound processes generated within the Kernel by the stress-ng tool, the increase in CPU Load and Memory Utilization by increasing number of users is 2.25% and 0.07% on average respectively. Then, the CPU bound processes based load is increased to 75% by running 3 processes with a ‘vm’ data of 1GB each and all the users which were previously connected are released. Again Figure 27 and Figure 26 gives an account of the respective CPU Load and Memory Utilization both in percentage and numerical values respectively. The same values are again monitored by increasing number of terminals having constant data traffic. As a result, the difference comes out to be 3% on the CPU Load and 0.03% on Memory Utilization while the stress-ng was generating 75% CPU bound processes load. Last but not the least, repeating the similar loading for all the cores present in SPGW using the stress-ng tool and increasing number of UEs, the difference comes out to be 2.16% and 0.22% on average respectively. The primary purpose of this load testing is to estimate the total number of users, the current deployed setup can hold and if needed how the resources should be enhanced in future. A logical conclusion would be that the CPU load and memory utilization increases with increase in number of UEs, no matter how much the base load created by CPU bound processes is.



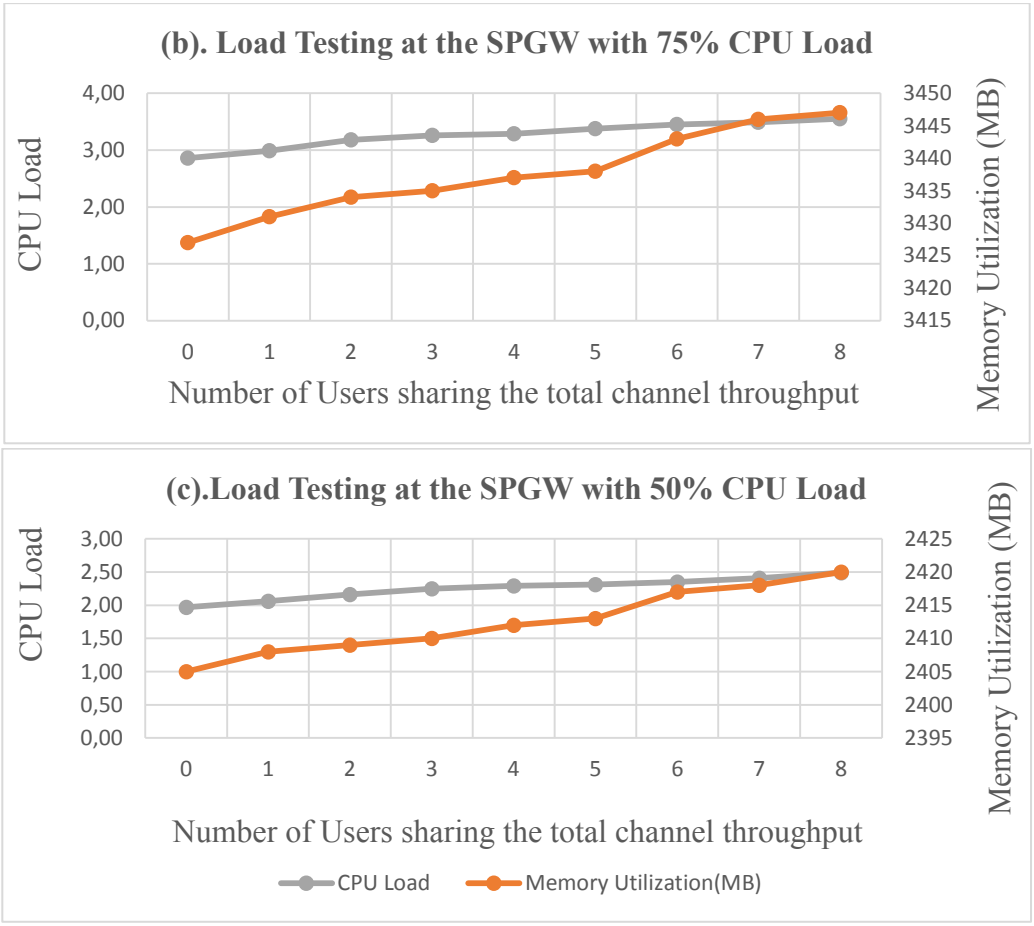
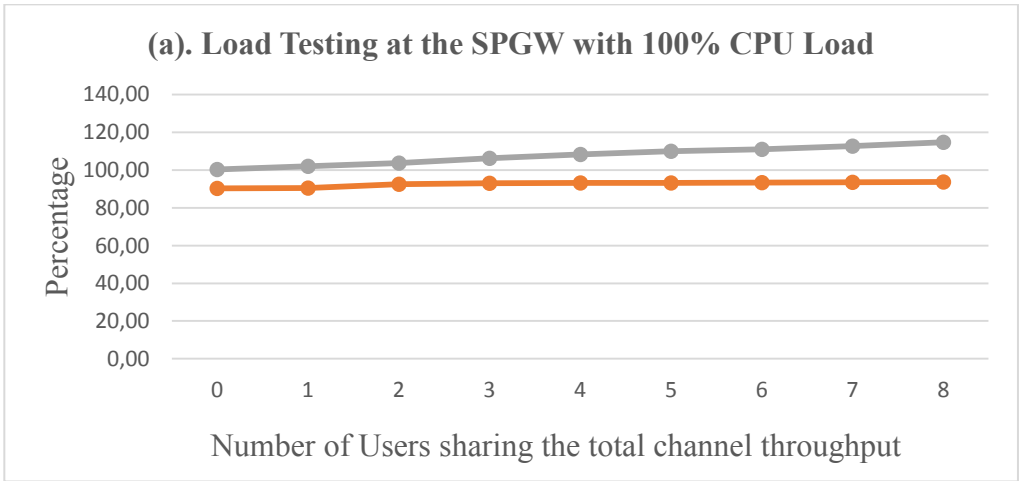


Figure 26. Load Testing at the SPGW. (a)The first plot represents CPU loading of 100% by using the stress-ng tool and 8 UE generating constant data traffic with its own Load as well. (b) The second plot represents 75% CPU Load by the kernel bound CPU Processes along with 8 UE. (c) The third plot depicts CPU Load of 50% by the CPU bound processes and 8 UE with a very small Load.



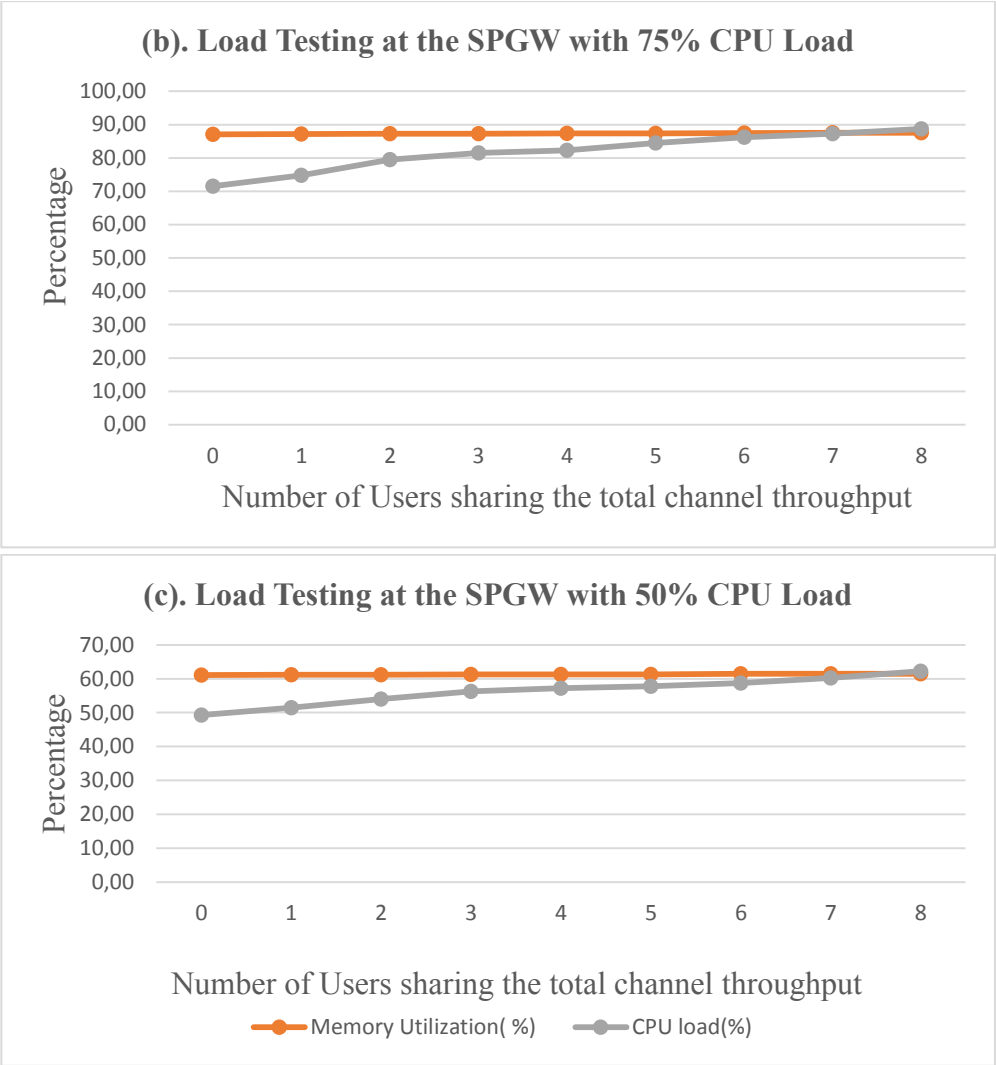


Figure 27. Load Testing at the SPGW with values in percentage. (a)The first plot represents CPU loading of 100% by using the stress-ng tool and 8 UE generating constant data traffic with its own Load as well. (b) The second plot represents 75% CPU Load by the kernel bound CPU Processes along with 8 UE. (c) The third plot depicts CPU Load of 50% by the CPU bound processes and 8 UE with a very small Load.

An estimate of the maximum number of UEs which could be entertained with the current resources at the SPGW is made based on some measurements which is given below:

- Difference of CPU load by increasing number of UEs when the stress-ng tool generates 50% load= 2.25% approx... (On average)
- Difference of CPU load by increasing number of UEs when the stress-ng tool generates 75% load= 3.0% approx... (On average)
- Difference of CPU load by increasing number of UEs when the stress-ng tool generates 100% load= 2.16% approx... (On average)
- So, Maximum number of users (with current CPU resources at the SPGW) =40-44 users on average.

5.2.2. Effect of Signaling Load on MME

Since the Control plane is equally important in LTE networks, a performance monitoring at a node within the Control plane path is requisite. MME being the authentication and authorization brain (along with the HSS) lies on 5GTN as a running node. Like other VMs, the resources allocated for MME on 5GTN is depicted in Table 3. The signaling in 5GTN with OpenEPC as a core network is illustrated in Figure 23 as dotted lines in chapter 4 of the thesis. Load Testing similar to the one performed on SPGW node is repeated on MME by increasing number of terminals as depicted in Figure 28. Though the signaling load is very small as compared to the data traffic load yet it uses more CPU and Memory resources within the MME. Figure 28 provides the testing scenario with increasing number of terminals and also varying the kernel CPU-bound processes. Initially, the CPU is half loaded by using stress-ng which generates 1GB vm processes and no UE is connected. With no signaling traffic and 50% CPU being utilized by stress-ng processes, the CPU load is 1.35 and the Memory Utilization is 1703MB. Then, we added a UE to have some signaling load added to the present CPU load which as a result increases the CPU load to 1.47 and the Memory utilization to 1704MB. Having analyzed these values, we keep on increasing the number of LTE terminals and with 8 UEs being added the CPU load is increased to 1.92 along with 1720MB of Memory Utilization. Since, the values are pretty random so taking an average of the difference in CPU load we came to a conclusion that every user adds 0.7 value to CPU load and around 2MB to Memory utilization.

Again, the whole experiment is repeated by enhancing the CPU-bound processes to load all the CPUs using stress-ng tool. When 8 terminals are added the CPU load and Memory Utilization increases up to 2.97 and 1729MB respectively. We can infer from the difference in values that with every UE signaling traffic, 0.6375 value is added to CPU load and 3.25MB to Memory Utilization. The difference on average comes out to be almost the same no matter how much loaded the CPU is. As each user adds almost 7% load to the system so the number of users which can be added with current resources on the MME is 14 only. Yet it is very strange to observe that, the memory utilization is almost the same on 50% and 100% load by stress-ng tool CPU bound processes. Since the SPGW can hold around 44 UEs (at most approx.) based on its resources but the QoS will be compromised because of less resources on the MME. As a matter of fact, the current EPC architecture involves a lot of control plane signaling among the EPC components which as a result generates a substantial amount of overhead in terms of bandwidth and processing delays [1]. Keeping the fact in mind that the forwarding traffic from the kernel incurs greater overhead [1], we observed some processes running within the MME which takes a large part of CPU. The MME has an avahi-daemon processes running at PID 438 (concerning DNS to register local IP addresses) which takes a lot CPU already as compared to the SPGW. Moreover, the CPU usage is more also because of the UEs attach and registration requests because of GTP tunnel setup request [14].This is the core reason of having a greater load on the MME as compared to SPGW. As a result, the number of cores and Memory should be enhanced on the MME so that with an average of 7% CPU load of every UE, the system can hold up to 44 users.

Assuming similar experiments on 4 core CPU in the MME VM, every UE will add 3% load on the VM and the VM can accommodate double the number of users i.e. 28. The MME current resources turned out to be the limiting factor in the current

implementation as the control plane is not capable to hold the traffic of 44 users (at most).

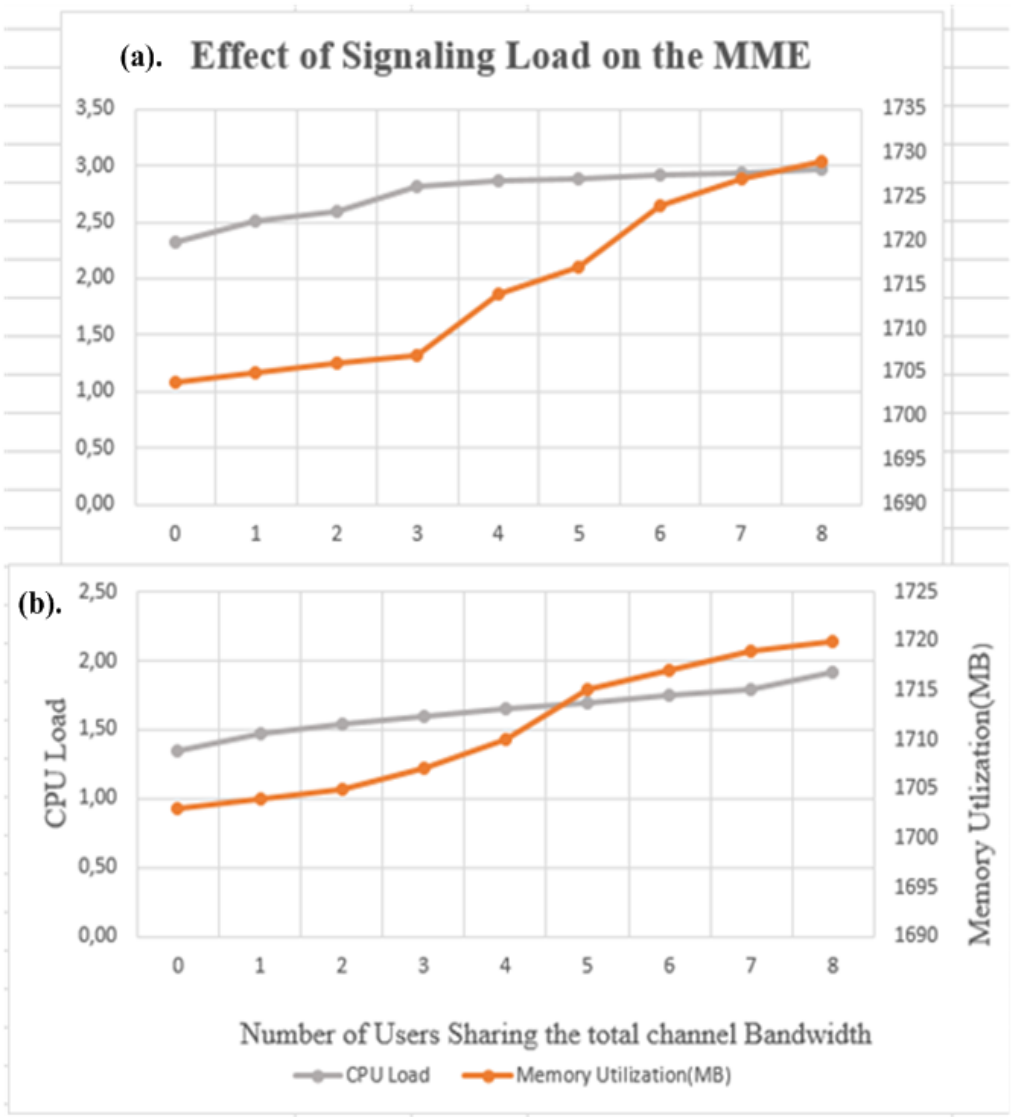


Figure 28. Signaling Load Testing at the MME. (a) The first plot represents CPU loading of 100% by using the stress-ng tool and the signaling load generated by 8 UE during attachment. (b) The second plot depicts CPU Load to 50% by kernel based CPU bound processes using stress-ng tool and along with the signaling Load of 8 UEs.

5.3. Iperf Configuration

Iperf is a tool used to find out the maximum bandwidth which could be achieved and can also monitor the quality of a network link[21]. Iperf test could be performed having an Iperf Server and Client mode enabled at different devices over the network using

TCP, UDP and SCTP streams[21]. Iperf uses the following script on the Client and Server side respectively:

- *Iperf –c <IP of the Iperf server> -t [total transmission time] -p [port to be connected to] –w [window size] –r [bidirectional test individually] -d [bidirectional test simultaneously] –M [Maximum segment TCP size] –i [time interval] –u [UDP stream]*
- *Iperf –s*

Since the channel bandwidth for Pico cell having 2.6GHz FDD spectrum, is set to 5MHz so the theoretical maximum throughput which could be achieved over the network with 64QAM is 25.2Mbps. Mathematical calculation for the theoretical maximum throughput is shown below:

- $71.4\mu s =$ symbol time duration in LTE
- $5\text{MHz} =$ Channel Bandwidth
- Since in case of 5MHz, 300 subcarriers are used
- In case of 64QAM, each symbol is allowed to carry 6bits
- Total bits carried by 300 subcarriers $= 1800$
- $\text{Throughput}_{\max} = {}^{1800}/_{71.4\mu s} = 25.2\text{Mbps}$
- Similarly, for 10MHz channel bandwidth $\text{Throughput}_{\max} = 50.\,\text{Mbps}$

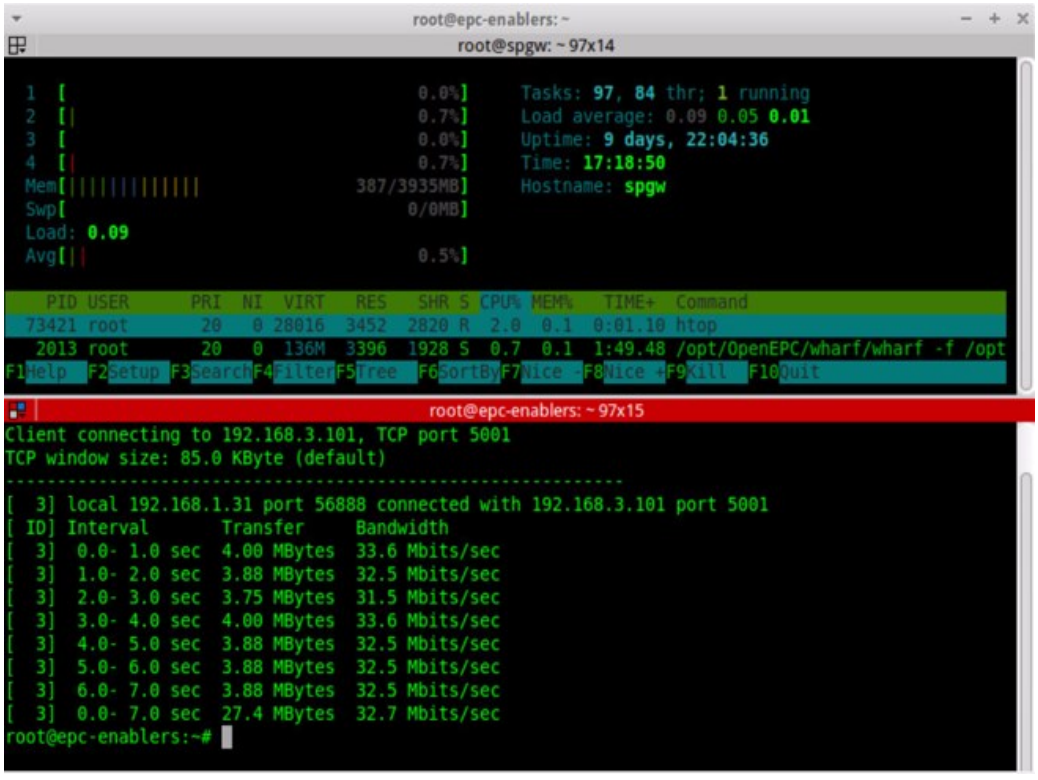


Figure 29. Iperf client with CPU Utilization.

The 5GTN performance could be monitored at a node within OpenEPC and the UE attached to the 2.6GHz Pico cell over LTE access network. The performance

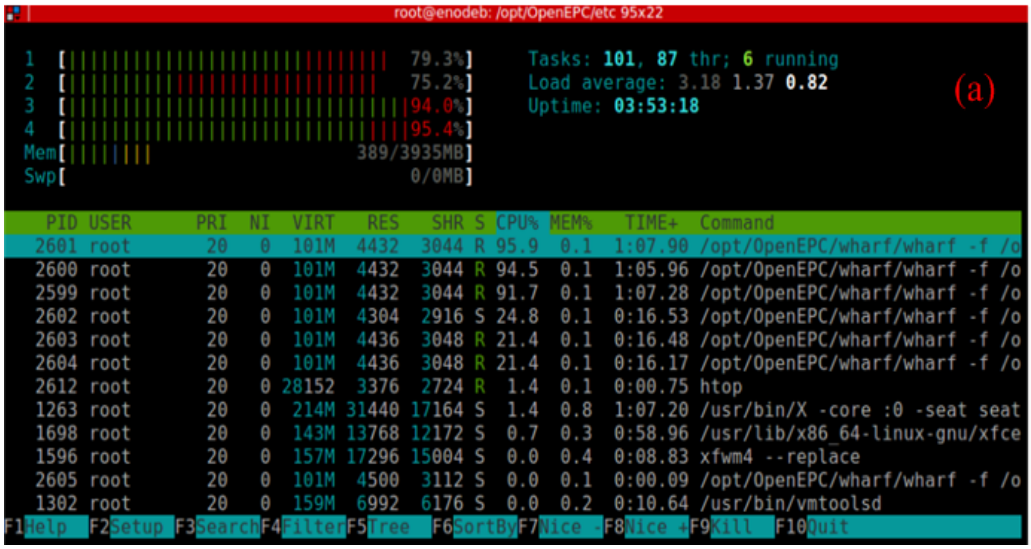
[21] Iperf Manual available online at https://iperf.fr/iperf-doc.php

monitoring scenario is such that the UE is initially configured as an Iperf Server and the EPC-Enablers virtual machine listens as an Iperf Client. The same process could then be repeated the other way around also. Meanwhile, CPU Load is being monitored by "htop" in SPGW. The script being used for both the Client and Server with the default window size is shown below:

- **Iperf Server:** Iperf –s –t 8 –i 1
- **Iperf Client:** Iperf –c 192.168.3.101 –t 8 –i 1

The EPC-Enablers serves as an Iperf Client and the UE connected over LTE as an Iperf Server as shown in Figure 29. The SPGW gives an account of the Load on CPU during the TCP stream is passed on from the core network all the way to the UE. Figure 29 exhibits that out of four cores in SPGW, merely 0.5% average CPU is utilized with an average load of 0.09 and the rest of the resources are still almost under-utilized.

The same test is then repeated by using the emulated client UE and eNodeB by setting up the client UE as an Iperf Server listening on TCP port 5001 and the EPC-enablers as an Iperf Client while monitoring the CPU usage on the SPGW machine as shown in Figure 30. Having a look at Figure 30 leaves an impression that using the emulated client and eNodeB on the server setup with optimization we have reached the throughput of 500-650Mbps, but the SPGW still has some capacity to be utilized, unfortunately the eNodeB emulator does not have such good performance and it is the limiting factor. Nevertheless, the system could still be scaled in case that there would be such need.



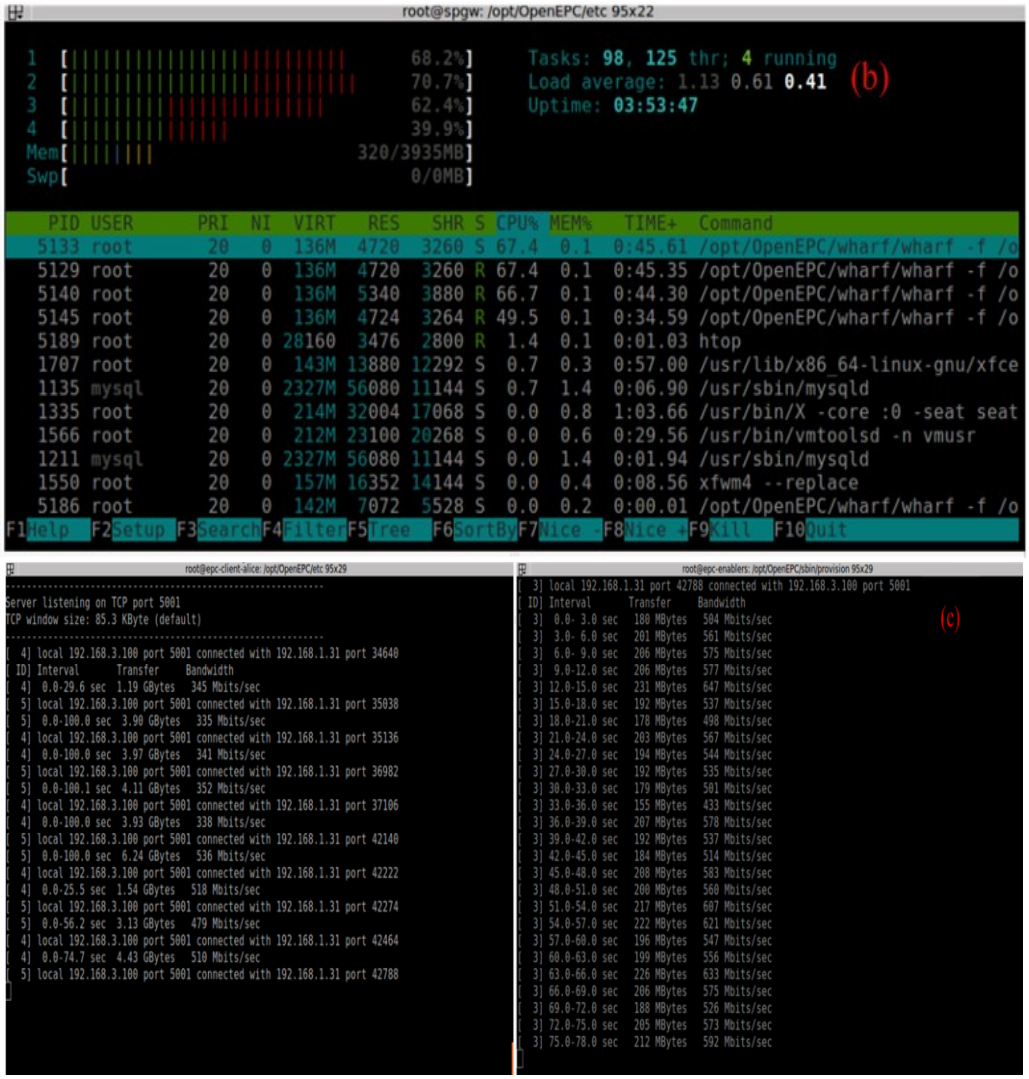


Figure 30. Iperf tests using emulated client and emulated eNodeB. (a) The first Figure shows the resource monitoring at the eNodeB using the 'htop' script. (b) The second Figure depicts the performance monitoring at the SPGW. (c) The third Figure shows the Iperf communication between the Client UE and EPC- Enablers VM.

5.3.1. Iperf Testing with 2.6GHz Pico Cell

In order to test OpenEPC in worst case scenario, all the vCPUs in EPC-Enablers and SPGW are being utilized to its maximum capacity. 'Stress-ng' tool, a workload generator to put a configurable amount of stress on the CPU and Memory[18], is used in the EPC-Enablers and SPGW machine using ***stress --vm-Number of virtual CPU to be loaded --vm-bytes 1G*** script. Using file 'stress.c', written in 'C', it has stressed all the four vCPUs leading to 100% CPU utilization as shown in Figure 32 below. It is important to note that 'N' corresponds to number of CPU cores. In case of EPC-Enablers we have 2 core as evident from Table 3, so we should write 2 instead of 4 in the 'stress script'. Contrary to the 25.2Mbits/s theoretical maximum throughput for a Pico cell with a channel bandwidth of 5MHz on a 2.6GHz spectrum, the Iperf test between a UE as an Iperf server and the EPC-Enablers machine, gives a maximum

throughput around 20.90Mbps with a very small CPU utilization. Figure 31 depicts the throughput graph for a TCP stream of 30 seconds between EPC-Enablers VM and a UE at 192.168.3.101 connected to LTE using OpenEPC as core network.

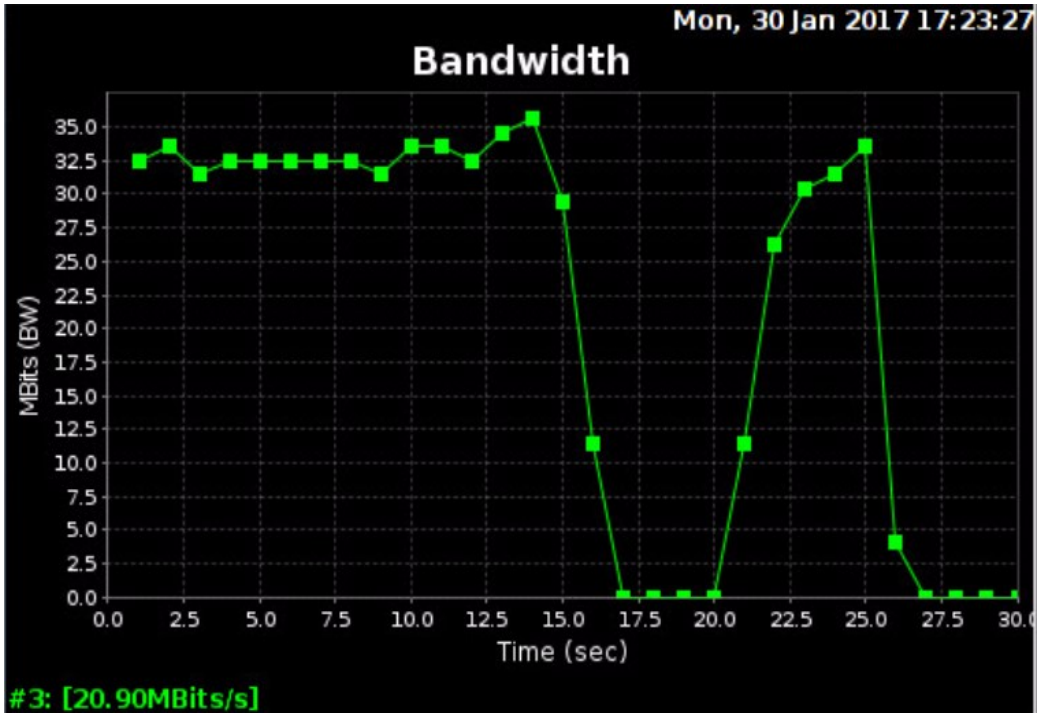


Figure 31. Maximum Throughput achieved at EPC-Enablers.

Since CPU utilization bottleneck results in performance degradation in virtualization platforms so testing OpenEPC while vCPUs within VMs are fully utilized will test the capability of the platform. Using the stress-ng tool, EPC-Enablers is fully stressed as displayed in Figure 32 and along with that the throughput shows the amount of degradation as compared to the one in Figure 31 i.e. it dropped from 20.90 to 18.40Mbps. When both the CPU cores are fully utilized, it gives a throughput of 18.40Mbps at the EPC/Enabler which is listening as an Iperf Client. In order to make the situation clearer Figure 33 provides us details with the similar Iperf test but with only two CPU cores under stress which primarily means CPU bound processes generated by the kernel to engage 50% of the resources within the VM.

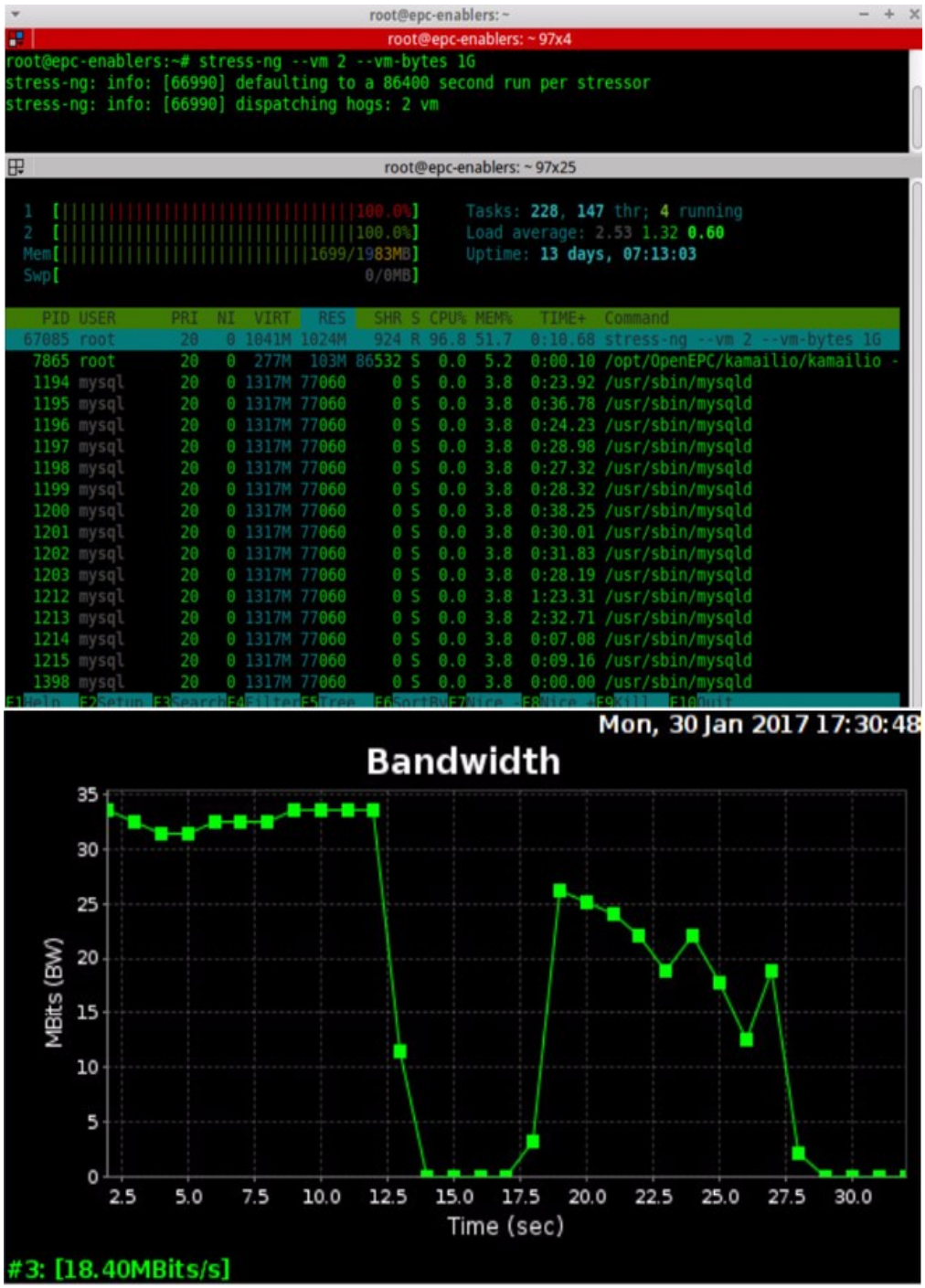


Figure 32. Throughput with 100% CPU Utilization at the EPC-Enablers VM. The Upper part shows the stress-ng tool generating CPU-bound kernel processes whereas the lower part depicts the maximum throughput achieved at the EPC-Enablers.

Contrary to the 100% CPU utilization scenario, when the CPU core in EPC-Enablers is 50% utilized as shown in Figure 33, the throughput is considerably increased to 19.30Mbps. Having performed these tests, it is quite evident now that the CPU utilization hinders LTE core network to perform efficiently and to use its maximum capacity.

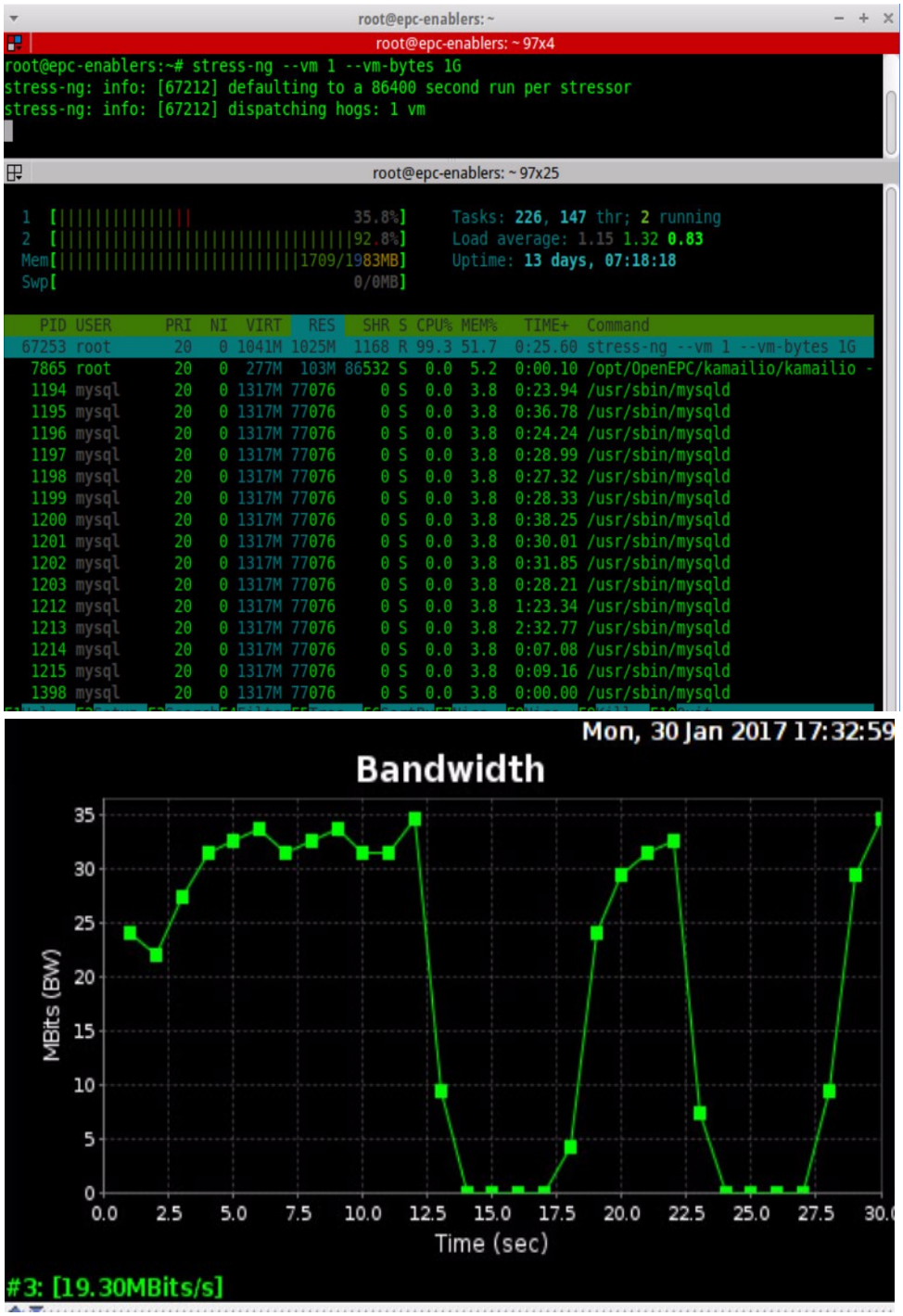


Figure 33. Throughput with 50% CPU Utilization. The upper portion of the Figure depicts half of the resources being utilized by the stress-ng tool in EPC-Enablers VM. The lower part illustrates the throughput achieved at the expense of 50% load and Memory Utilization.

Similarly, the next node in the Data plane path is the SPGW which communicates with the eNodeB via gtp tunnel, Iperf tests were performed by creating an Iperf server at the UE and SPGW being the Iperf Client. Initially, the maximum achievable throughput is monitored and then the throughput is taken by stressing all the four cores in the SPGW VM. The results are not that promising as compared to that of EPC-

Enablers yet we come up with a rule of thumb that the less cores stressed the more throughput we get. Every time throughput gives a different value within the specified range in the figure yet the maximum and minimum values stays the same.

5.3.2. Iperf Testing within OpenEPC Nodes

Since, we already have monitored the throughput between the UE and the gateways, we also need to monitor the throughput between two nodes within OpenEPC. So, an Iperf Server is created at the EPC-Enablers machine and the SPGW serves as an Iperf Client.

Where,

- *Iperf –s (EPC-Enablers)*
- *Iperf –c 192.168.1.70 –t 30 –i 1 –f m –r -m (SPGW)*

Using Iperf for a session of 30 seconds, the throughput we got on both Client and Server sides could be observed in Figure 34. The average throughput between these two OpenEPC nodes turned out to be 2.51Gbps with very less CPU and memory utilization.

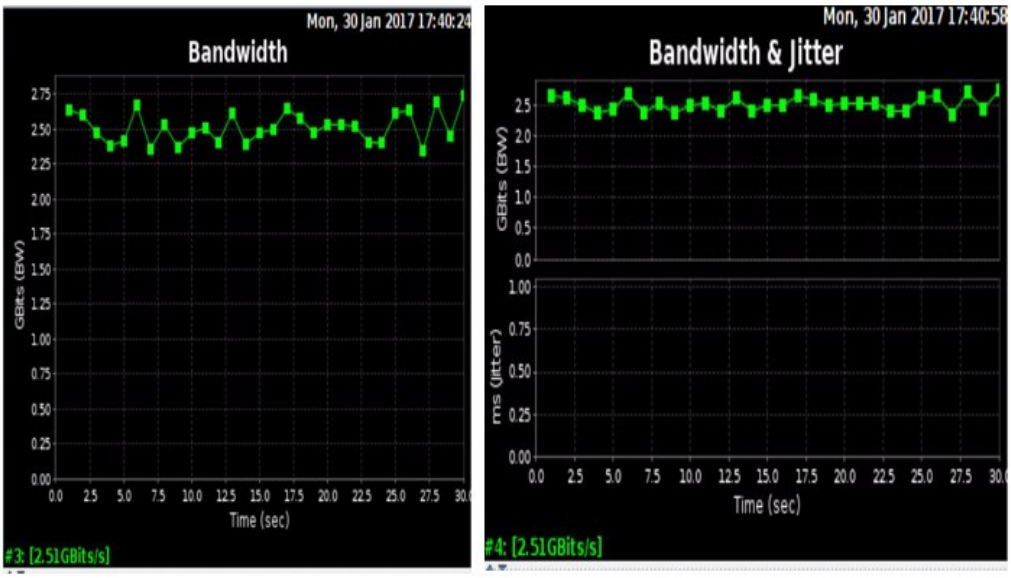


Figure 34. Throughput in SPGW and EPC-Enablers.

EPC-Enablers and SPGW has 2 and 4 vCPU cores respectively, which in case utilized to its maximum capacity leads to performance degrading. Both the VMs are fully stressed with 100% CPU being utilized, CPU status and the stress-ng script both could be seen in Figure 35.

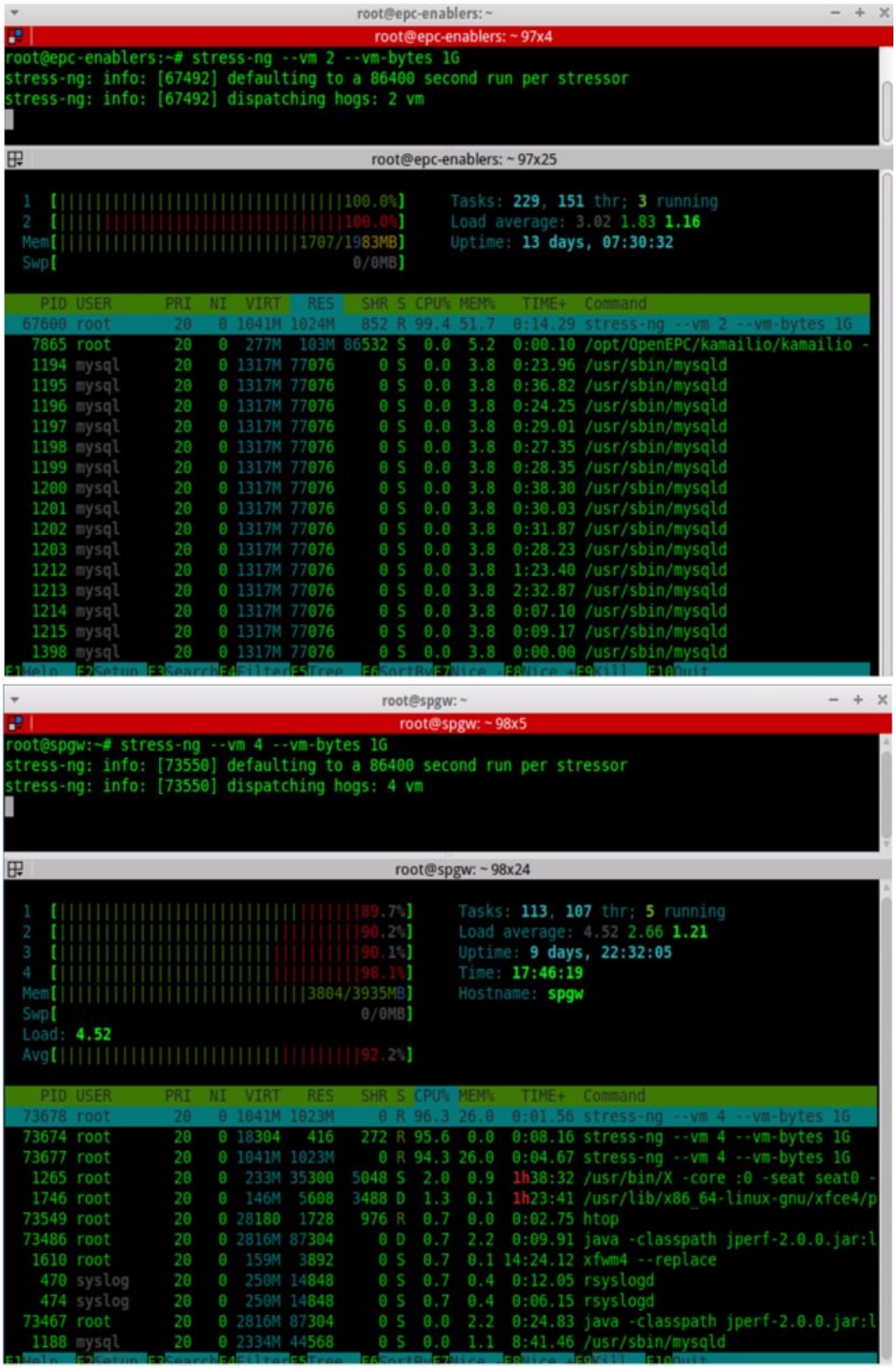


Figure 35. Virtual CPUs being fully stressed in both SPGW and EPC-Enablers. Stress-ng tool generates CPU bound processes to consume all the CPU and Memory resources. The Upper part of Figure represents the EPC-Enablers VM and the lower one is for SPGW.

When the same tests were repeated between EPC-Enablers and SPGW with 100% CPU utilization, the throughput gets reduced i.e. 2.21Gbps. Though the effect is not major but still the maximum CPU utilization will lead to performance degradation which is quite evident from Figure 36.

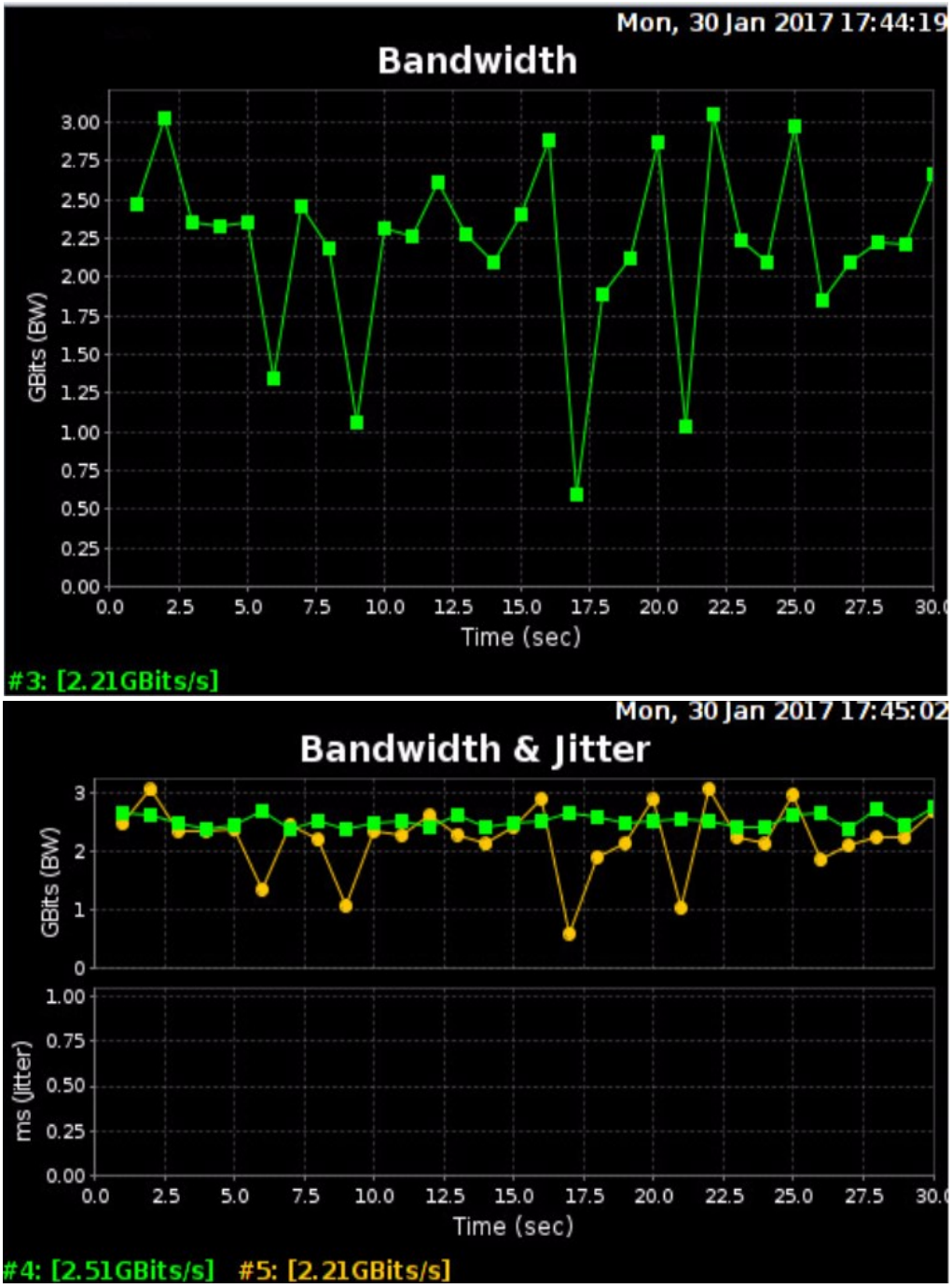


Figure 36. Throughput while CPU being fully stressed. The upper part of the Figure is the throughput achieved at the SPGW acting as an Iperf client and the lower part shows EPC-Enablers' Throughput which acts here as an Iperf server.

6. DISCUSSION

5G, which is mostly being discussed nowadays in Telecommunications world, is expected to bring lots of changes in the current infrastructure. Massive MIMO, MTC, MEC, IoT etc. comes in mind, every time 5G is being discussed. But one thing is very obvious that it should have a less network latency, more throughput and a more scalable network than the existing ones. To reach these expected goals, one of the major stream discussion has always been the complete split of Control and Data planes in LTE networks. The idea would be to have a virtualized Cloud based Control plane and then different data planes are orchestrated on demand by a centralized software based orchestrator which could be implemented as a virtual instance on the network by using ETSI'S NFV MANO. The way we approached 5G in this thesis work is the amalgamation of IT virtualization to standard LTE networks, to create a testbed which has the capability to support SDN and NFV functionalities. OpenEPC being already integrated within the testbed, virtualized the LTE core network elements. So, the next step would be to use OpenSourceIMS for extending the network to support IMS functionalities and then Open Baton for network management and orchestration. Open Baton which is in compliance with ETSI frame work for MANO enables virtual network service deployments on an NFV infrastructure[22], making the network orchestrateable. Since Open Baton requires Open Stack for which the whole testbed would soon be implemented in CSC-IT Centre for Science. Where, CSC is Finnish center of ICT that provides ICT services to Universities and research centers in Finland primarily providing OpenStack Cloud[23].

The thesis work primarily includes integrating OpenEPC within 5GTN and then having a performance evaluation on the virtual testbed while using physical LTE RAN. So, for this reason load testing is performed at different nodes within OpenEPC both on the Control and Data plane. Each UE being connected to LTE network and having some constant data traffic consumes 7% of the total CPU, generating 0.7 CPU load on the MME and 3% of the total load in the SPGW. We came to a conclusion that with the current resources the SPGW can handle data traffic from 40-44 LTE UEs approximately but the MME is capable to handle signaling from 14 UEs at a time approximately owing to the limited resources on the MME VM. In order to avoid such a limitation, the MME resources should be scaled accordingly.

Iperf tests is being conducted at different OpenEPC nodes to find the performance throughput. The test is performed between SPGW and a UE connected over 2.6GHz FDD LTE, the maximum throughput achieved was 20.90Mbps which is pretty close to the theoretical maximum for 5MHz channel bandwidth. Since, OpenEPC has an emulation of eNodeB and client UE which are also running as virtual instances in 5GTN, an Iperf test performed between EPC-Enablers and Client UE yields a throughput of 500-650 Mbps. Moreover, the test is also performed within OpenEPC nodes i.e. between EPC-Enablers and SPGW, the maximum achieved throughput was 2.51Gbps. Having said that, all the research questions are answered which were mentioned in the Objectives and scope of the 'Introduction' chapter. Virtualization apparently results in some performance degradation bit could be overcome by using enough resources and making the infrastructure scalable. Apart from the 5GTN

[22] Open Baton: an open source reference implementation of the ETSI MANO Available online at https://openbaton.github.io/index.html#about

[23] CSC-IT Centre For Science Available at https://www.csc.fi/csc

testbed, there are lots of similar functional testbeds for research purposes around the globe. Several infrastructures are listed on XiPi+ webpage which can have inter collaboration to achieve their expected targets[24].

There has been ample of research work on restructuring the LTE EPC by introducing the concepts of NFV and SDN. For instance, the research work in [1] primarily projected two approaches i.e. SDN and NFV based LTE EPC deployment respectively. The SDN based EPC deployment comprises of Control and Data plane splitting for the gateways. Forwarding packets based on the controller instructions, the Data plane of the gateways are being realized as OpenFlow switches. It is very interesting that the eNodeB also has an SDN switch which has the capability to divert traffic to EPC switches by taking instructions from the controller. Furthermore, the NFV based deployment comprises of software modules of the core network functions. These modules are running in individual VMs on a private cloud and there is a RAN-simulator module also which can combine functionality of SCTP clients. Both the deployments were analyzed individually in terms of throughput, CPU utilization and latency. Having separate Control and Data planes, the SDN based deployment, has the SDN controller which has control functions running as applications beneath. While the NFV deployment has virtualized the core network components into VMs running on data centres. It Finalized the work with conclusions that the SDN based EPC is better while handling large amount of data traffic and the NFV based EPC is better at handling large signaling load. [1]

The research work in [15] is a joint effort of different researchers utilizing a testbed at Aalto University to analyse the usage of NFV and SDN. The primary idea is again to split the LTE Control and Data Plane where SDN will provide with a tool for network orchestration and NFV will migrate the network components in a cloud. More specifically, the testbed comprises of Nokia RAN (eNodeB), an MPLS switch made by Coriant having OpenFlow capability and EXFO based traffic monitoring probes. 'Ryu' serves the purpose of an SDN controller and the S/PGW is an open source EPC SAE Gateway. It is important to note that all the software components are running in a data center at Aalto University. Using the testbed, three scenarios were being implemented to showcase the basic requirements for seamless migration. The EXFO probes provides constant monitoring to provide information regarding the network. The experimentation is performed based on three use cases which demonstrates the requirements for provisioning and optimization, cost reduction and security respectively. After several performance tests, this research work concludes with the fact that SDN and NFV is capable to address several needs of 5G. Though SDN made the optimal usage of resources with less overhead yet there are several downsides like reliability, robustness and high latency (while moving VM with network elements owing to hardware failure). [15]

There are several testbeds with complete infrastructure created around OpenEPC as a 3GPP EPC and could be utilized form outside once access is granted e.g. 'emulab' by PhantomNet[25]. The research work in [10] projected three deployment scenarios i.e. OpenEPC VM, Dockers VM and Custom VM. OpenEPC depicts performance improvement compared to the deployment based on other VMs [10]. The research

[24] A marketplace for stakeholders of Next Generation Internet Research and Experimentation Available at: https://www.xipi.eu/Infrastructures

[25] 'emulab' total network testbed available at: https://wiki.emulab.net/wiki/phantomnet

work also comprises of container based deployment of OpenEPC which if made completely automated would transform EPC core to a plug and play process [10].

The performance evaluation done in this thesis work in case of OpenEPC in 5GTN in terms of throughput compared to the theoretical maximum and expected 5G Data rates makes the research work unique compared to the above mentioned research works. Since the evaluation is performed with 2.6GHz Pico cells only, integration of 3.5GHz TDD within the testbed in general and OpenEPC in particular is requisite. As a future work, the CPE device as a UE category (supporting band 42) could be connected to 3.5GHz TDD using OpenEPC core. A performance evaluation by comparing the results from this implementation to the one which we have already done in this thesis, would lead us to more empirical conclusions. Moreover, the testbed in future also requires load testing using traffic monitoring tools, for which 5GTN most likely would use EXFO tools. As a result, the OpenEPC deployment in 5GTN as a proof of concept for NFV is still not complete as we lack the MANO platform. So, the testbed at this stage is just an infrastructure where we have virtual core network components being bridged to RAN over 5GvLAN. So, a lot of work is still need to be done to enhance the infrastructure to actually practice NFV. As a future prospect, OpenBaton is expected to be integrated within the network which would serve as an SDN controller for the network.

7. SUMMARY

NFV will bring a revolution in Telecommunications industry without performance degradation. The integration of OpenEPC in 5GTN made the testbed efficient to support the possibility of having any NFV and SDN based use cases. The OpenBaton and Open IMS would be the next functionalities to be added to the testbed to have MANO and IMS support respectively. Yet the performance evaluation of the testbed leads us to several conclusions. Since virtualization techniques has always been criticized for CPU utilization bottlenecks, the testbed is evaluated based on CPU and Memory utilization. The SPGW being part of the data plane path is efficient enough based on its resources to pass the data traffic of at most 44 users approx. all the way to the UE and vice versa. The increase in number of UEs having constant traffic ultimately increases the CPU load. Most importantly, the CPU load and Memory utilization increases with increasing number of terminals no matter how much the base load in the VM itself is. The virtual resources are not optimal at the SPGW yet it is efficient enough for a testbed which will be used for research purposes. On the other hand, in the MME being part of the control plane, the existing resources leads to a bottleneck scenario. Keeping the fact in mind that the LTE EPC has a huge signaling overhead which leads to processing delays, the MME in our testbed has signaling which takes a large CPU initially leading to greater CPU utilization on the MME as compared to SPGW. Upon attachment and requesting for GTP-C tunnels, the CPU load increases which is also a major reason of having greater load on the MME. As far as the throughput is concerned, the values achieved by Iperf testing is pretty close to the theoretical maximum for 5MHz channel bandwidth with 64QAM on a 2.6GHz FDD LTE Spectrum. The throughput is compromised by having too much base load in individual VMs in a way that the more CPU bound processes we have in the VM, the less throughput we get.

The testbed is efficient for NFV and SDN, as it has compatibility with Open Baton and OpenSourceIMS. OpenEPC has already been used around the world in different testbeds supporting several use cases. Thus, 5GTN could be used for testing use cases as the platform is stable, efficient and scalable.

8. REFERENCES

[1] A. Jain, S. N S, S. K. Lohani and M. Vutukuru, “A Comparison of SDN and NFV for Re-designing the LTE Packet Core,” 2016.

[2] H. Hawilo, A. Shami, M. Mirahmadi and R. Asal, “NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC),” *IEEE Network,* vol. 28, no. 6, pp. 18-26, 2014.

[3] J. Doherty, SDN and NFV Simplified: A Visual Guide to Understanding Software Defined Networks and Network Function Virtualization, Donnelley in Kendallville, Indiana: Addison-Wesley Professional , 2016.

[4] R. Jain and S. Paul, “Network Virtualization and Software Defined Networking for Cloud Computing: A Survey,” *IEEE Communications Magazine ,* vol. 51, no. 11, pp. 24-31, 2013.

[5] J. Sugerman, G. Venkitachalam and . B.-H. Lim, “Virtualizing I/O Devices on VMware Workstation’s,” in *Proceedings of the 2001 USENIX Annual Technical Conference*, Boston, Massachussetts,USA, 2001.

[6] R. Mijumbi, J. Serrat, J.-l. Gorricho, S. Latre, M. Charalambides and D. Lopez, “Management and orchestration challenges in network functions virtualization,” *IEEE Communications Magazine,* vol. 54, no. 1, pp. 98-105, 2016.

[7] K. Gray and T. Nadeau, Network Function Virtualization, Cambridge, United States: Todd Green, 2016.

[8] A. Skopljak-Ramovic and S. Pivac, “The challenge of implementation of long term evolution / system architecture evolution (LTE/SAE),” in *MELECON 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference*, Malta, 2010.

[9] M. Corici, F. Gouveia, T. Magedanz and D. Vingarzan, “OpenEPC: A Technical Infrastructure for Early Prototyping of NGMN Testbeds,” in *Testbeds and Research Infrastructures. Development of Networks and Communities. TridentCom* , Berlin, 2010.

[10] J. Fontenla-Gonzalez, C. Perez-Garrido, F. Gil-Castineira, F. J. Gonzalez-Castano and C. Giraldo-Rodriguez, “Lightweight container-based OpenEPC deployment and its evaluation,” in *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, Kansas City, 2016.

[11] M. Corici, C. Mihai, S. Dana, V. Dragos, V. Valentin and W. Lukas, “Integrating off-the-shelf 3GPP access networks in the OpenEPC software toolkit: Realizing cost-efficient and complete small-scale operator testbeds,” in *Globecom Workshops (GC Wkshps), 2012 IEEE*, Anaheim, California, 2012.

[12] M. Corici, D. Vingarzan and T. Magedanz, “3GPP Evolved Packet Core - the Mass Wireless Broadband all-IP architecture,” in *Telecommunications: The Infrastructure for the 21st Century (WTC), 2010*, Vienna, Austria, 2010.

[13] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann and E.-D. Schmidt, “A Virtual SDN-enabled LTE EPC Architecture: a case study for S-/P-Gateways functions,” in *IEEE Software Defined Networks for Future Networks and Services (SDN4FNS)*, Trento,Italy, 2013.

[14] S. K. Lohani, “Building and benchmarking SDN-based LTE EPC Gateways,” Department of Computer Science and Engineering Indian Institute of Technology, Bombay, Mumbai, 2015.

[15] J. Costa-Requena, J. Llorente Santos, V. Ferrer Guasch, K. Ahokas, G. Premsankar, S. Luukkainen, I. Ahmad, M. Liyanage, M. Ylianttila, O. López Pérez, . M. Uriarte Itzazelaia and E. Montes de Oca, “SDN and NFV integration in Generalized Mobile Network Architecture,” in *European Conference on Networks and Communications (EuCNC)*, Paris, France, 2015.

[16] M. Olsson, C. Mulligan, M. Olsson, S. Rommer, C. Mulligan, S. Sultana and L. Frid, SAE and the Evolved Packet Core, Academic Press, 2009.