



**UNIVERSIDAD
DE GRANADA**

TRABAJO FIN DE GRADO
INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Emulación de redes móviles 4G usando OpenAirInterface

Autor

Manuel Díez Galiano

Director

Jorge Navarro Ortiz



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, julio de 2020



**UNIVERSIDAD
DE GRANADA**

Emulación de redes móviles 4G usando OpenAirInterface

Autor

Manuel Díez Galiano

Director

Jorge Navarro Ortiz

Emulación de redes móviles 4G usando OpenAirInterface

Manuel Díez Galiano

Palabras clave: OpenAirInterface, SDN, eNB, UE, OAISIM, EPC, E-UTRAN, HSS, MME, SPGW, iperf, LTE, 3GPP, Oracle VM VirtualBox.

Resumen

El inicio de las redes móviles data de 1894 con la invención del telégrafo inalámbrico, aunque prácticamente nadie tenía acceso a este tipo de equipos. En la década de los 80's aparece la primera generación (1G) basada en comunicaciones analógicas. La segunda generación trajo consigo los primeros sistemas móviles digitales y supuso una gran expansión y accesibilidad a la población de los servicios de comunicaciones móviles de voz y datos, además de la posibilidad de aumentar la movilidad entre países con la implementación del *roaming* internacional. 3G supuso el comienzo de los sistemas móviles de banda ancha.

A día de hoy, las redes móviles que utilizan el estándar LTE (*Long Term Evolution*) de 3GPP (*3rd Generation Partnership Project*) dan cobertura al 95% de la población en países desarrollados y al 70% de la población en países en vías de desarrollo. Estas redes se encuentran a pleno rendimiento, siendo las que a día de hoy se comercializan con mayores prestaciones ante el incipiente despliegue de 5G. Proporcionan alta velocidad, alta calidad, alta capacidad, seguridad y servicios de bajo coste para servicios de voz y datos, multimedia e internet a través de IP (*Internet Protocol*).

Este proyecto se centra en el estándar LTE. Gracias al uso de un dispositivo *Software Defined Radio* (SDR), se pretende crear una red basada en este estándar. Para ello, se hará uso del software *Open Air Interface* y se crearán dos máquinas virtuales creadas con el software *Oracle VM VirtualBox*. Una de ellas implementará la red de acceso E-UTRAN (*Evolved Universal Terrestrial Radio Access Network*) del estándar LTE. La otra implementará la red troncal EPC (*Evolved Packet Core*). Para finalizar el proyecto, una vez creada y configurada correctamente el conjunto de la arquitectura, se llevará a cabo una evaluación del rendimiento de la red.

Debido a la pandemia mundial sufrida por el Covid-19 el proyecto se realizará mediante emulación de la red al completo en lugar del uso del SDR para la parte de red de distribución. La emulación se hará con el paquete *Open Air Interface System Emulation* (OAISIM) de *Open Air Interface*.

4G mobile network emulation using OpenAirInterface

Manuel Díez Galiano

Keywords: OpenAirInterface, SDN, eNB, UE, OAISIM, EPC, E-UTRAN, HSS, MME, SPGW, iperf, LTE, 3GPP, Oracle VM VirtualBox.

Abstract

The beginning of mobile networks dates back to 1894 with the invention of the wireless telegraph, although virtually no one had access to such equipment. In the 1980s, the first generation (1G) based on analog communications appeared. The second generation brought with it the first digital mobile systems and caused a great expansion and accessibility to mobile voice and data communication services, as well as the possibility of increasing mobility between countries with the implementation of international roaming. 3G was the beginning of mobile broadband systems.

Today, mobile networks using the 3GPP (*3rd Generation Partnership Project*) LTE (*Long Term Evolution*) standard cover 95 % of the population in developed countries and 70 % of the population in developing countries. These networks are fully deployed, being those that today are marketed with greater performance. They provide high speed, high quality, high capacity, security and low-cost services for voice and data services, multimedia and Internet over IP (*Internet Protocol*).

This work is focused on LTE standard. Thanks to the usage of an *Software Defined Radio* (SDR) device, we will build a network prototype based on this standard. To do this, we will use *Open Air Interface* software and will create two virtual machines using *Oracle VM VirtualBox* software. One of them will take the role of the access network, i.e. the E-UTRAN (*Evolved Universal Terrestrial Radio Access Network*). The other will implement the core network, i.e. the EPC (*Evolved Packet Core*). To complete this project, once the LTE network prototype is successfully created and configured, a performance assessment will be performed.

Due to the global pandemic of Covid-19, this project will be carried out by emulating the entire network instead of using an SDR device for the distribution network. The emulation will be done with the package *Open Air Interface System Emulation* (OAISIM) included in Open Air Interface software.

Yo, **Manuel Díez Galiano**, alumno del Grado en Ingeniería de Tecnologías de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Manuel Díez Galiano

Granada a 7 de julio de 2020 .

D. **Jorge Navarro Ortiz**, Profesor Titular de Universidad del Área de Ingeniería Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Emulación de redes móviles 4G usando OpenAirInterface***, ha sido realizado bajo su supervisión por **Manuel Díez Galiano**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 7 de julio de 2020.

EL director:

Jorge Navarro Ortiz

Agradecimientos

Con la entrega de este proyecto pongo fin a una de las etapas más importantes de mi vida. Estoy muy contento por haber conseguido esta meta que supone el principio de mi carrera como profesional de un sector que me apasiona. También me siento orgulloso de haberlo conseguido después de tanto esfuerzo y dedicación durante estos años. Sé que no he estado sólo en este camino y por eso me gustaría agradecerse a quienes han estado presentes.

A mis padres, por el gran esfuerzo que han realizado para que pudiera conseguir ser quien soy hoy y todo el apoyo que me han dado durante toda mi vida. Por creer en mí en todo momento, incluso cuando ni yo mismo lo hacía. Espero que llegéis a estar tan orgullosos de mí como yo lo estoy de vosotros.

A mi hermana, por ser una hermana mayor ejemplar y haber puesto siempre el listón de la familia tan alto. Siempre has sido todo un ejemplo a seguir para mí.

A mi compañera de vida, de penas y de alegrías, de tantos buenos momentos vividos juntos. Siempre has estado a mi lado y me has apoyado cuando te he necesitado. Me siento muy afortunado de que llegaras a mi vida para quedarte Belén, te quiero.

A toda mi familia y en especial a mis abuelas, sois un tesoro que tengo y me hace muy feliz y sentirme orgulloso lo mucho que os alegráis de mis logros.

A mis abuelos, que sé que desde allá donde estéis me habéis estado apoyando y me habéis visto conseguir esta meta, siempre os llevo en mi corazón.

A todos mis compañeros con los que he compartido este grado, por todos esos momentos vividos durante esta etapa y haber hecho más llevadero los días de clase, prácticas y exámenes.

A mis amigos, porque desde la infancia hemos vivido grandes momentos

juntos y habéis sido un gran apoyo. Da igual cuanto pase sin vernos, cuando lo hacemos es como si no hubiera pasado el tiempo.

A mi tutor, Jorge Navarro, por motivarme desde el comienzo de este proyecto y haber sido mi guía durante su realización. Por dedicar tanto tiempo a conseguir la elaboración de este proyecto y resolver mis dudas sin importar horarios ni días de la semana.

Muchísimas gracias a todos.

Índice general

Índice de figuras	v
Índice de tablas	vii
Siglas	ix
1. Introducción	1
1.1. Contexto	1
1.1.1. LTE en España	2
1.1.2. 5G en España	4
1.2. Motivación	4
1.3. Objetivos	4
1.3.1. Objetivos del escenario del proyecto inicial	5
1.3.2. Objetivos del escenario del proyecto adaptado	5
1.4. Estructura de la memoria	6
2. Estado del arte	9
2.1. srsLTE	9
2.2. OpenLTE	10
2.3. Amari LTE 100	10
2.4. Open Air Interface	11
2.5. Conclusiones y solución elegida	12
3. Planificación y presupuesto	13
3.1. Planificación y organización de las fases del proyecto	13
3.2. Inventario y presupuesto	16
3.2.1. Recursos hardware	17
3.2.2. Recursos software	23
3.2.3. Recursos humanos	24
3.2.4. Presupuesto total	25
3.3. Modificación de la planificación, inventario y presupuesto. . .	25
3.3.1. Modificación de la planificación	25
3.3.2. Modificación del inventario y presupuesto	25

4. Fundamentos Teóricos	29
4.1. LTE	29
4.1.1. Estructura de la red LTE.	30
4.1.2. Arquitectura de protocolos	34
4.1.3. Canales MAC	36
4.1.4. Canales físicos	40
4.1.5. Tecnologías usadas en LTE	41
4.2. Open Air Interface	43
4.2.1. Objetivos perseguidos con su creación de código abierto	43
4.2.2. OpenAirInterface Software Alliance	44
4.2.3. EURECOM	45
4.2.4. Componentes del software OAI	46
5. Implementación	49
5.1. Creación de las máquinas virtuales	49
5.2. Instalación	53
5.2.1. Instalación de preliminares	53
5.2.2. Instalación de paquetes necesarios	56
5.3. Configuración	58
5.3.1. Configuración OAISIM	58
5.3.2. Configuración MME	60
5.3.3. Configuración HSS	64
5.3.4. Configuración SPGW	68
5.4. Compilación y puesta en marcha de todo el escenario	69
6. Evaluación del escenario	73
6.1. Prueba de rendimiento de la red	73
6.1.1. Iperf	73
6.1.2. Resultados obtenidos	75
6.1.3. Análisis de resultados	76
6.2. Señalización	78
6.2.1. Señalización de la conexión eNB	79
6.2.2. Señalización de la conexión UE	80
6.2.3. Señalización de autenticación y seguridad	80
7. Conclusiones	85
7.1. Conclusión	85
7.2. Líneas futuras	86
7.3. Valoración personal	87
Bibliografía	91
Apéndices	93

A. Archivos de configuración de EPC	93
A.1. hss.conf	93
A.2. hss_fd.conf	95
A.3. mme.conf	98
A.4. mme_fd.conf	104
A.5. acl.conf	107
A.6. spgw.conf	108
B. Archivos de configuración de OAISIM	113
B.1. enb.band7.tm1.usrpb210.conf	113

Índice de figuras

1.1. Logo 3GPP [1]	1
1.2. Tasa de datos máxima frente a movilidad en las diferentes generaciones [2]	2
1.3. Mapa porcentual de cobertura LTE por provincias en España	3
3.1. Diagrama de Grantt del proyecto	15
3.2. Asus X554LD	17
3.3. Dell Vostro 270	18
3.4. Fujitsu Celsius M770X	19
3.5. Monitor ASUS VZ239HE	19
3.6. Terminal móvil Samsung Galaxy A40	20
3.7. USRP National Instrument mo.2901	21
3.8. Antena Ettus VERT2450	21
3.9. sysmoUSIM-SJS1 SIM + USIM Card	22
4.1. Arquitectura de red LTE [3]	30
4.2. Arquitectura de EPC	31
4.3. Arquitectura de E-UTRAN	32
4.4. Aspecto físico eNB (fabricante Khomp) [4]	33
4.5. Pila protocolos plano de control y de usuario [5]	34
4.6. Segmentación/compresión RLC [2]	35
4.7. Mapa de canales en enlace descendente [2]	37
4.8. Mapa de canales en enlace ascendente [2]	39
4.9. Espectro señal OFDM con cuatro canales [6]	42
4.10. Proceso SC-FDMA [7]	43
4.11. Logo Open Air Interface [8]	44
4.12. Logo EURECOM [9]	46
5.1. Estructura lógica de la red del escenario.	50
5.2. Prueba de conexión OAISIM	51
5.3. Prueba de conexión EPC	52
5.4. Interfaces de red OAISIM	52
5.5. Interfaces de red EPC	53
5.6. Hostname EPC	58

5.7. Listado de tablas base de datos de OAI.	65
5.8. Tabla mmeidentify de la base de datos de OAI.	66
5.9. Interfaz generada en EPC para el uso de OAI.	70
5.10. Interfaz generada en OAISIM para el uso de OAI.	71
5.11. Prueba de conexión desde OAISIM hacia EPC mediante la interfaz de uso de OAI.	72
6.1. Curva de rendimiento del ancho de banda en nuestra red LTE para nuestros 3 tipos de configuración del ancho del BW de eNB	78
6.2. Mensajes de señalización entre nuestras VM.	79
6.3. Información contenida en el mensaje S1SetupRequest	79
6.4. Información contenida en el mensaje S1SetupResponse	80
6.5. Información contenida en el mensaje InitialUEMessage	81
6.6. Mensajes de señalización de autenticación y seguridad	81
6.7. Información contenida en el mensaje Authentication Request	81
6.8. Información contenida en el mensaje Authentication Response	82
6.9. Información contenida en el mensaje InitialContextSetup	83

Índice de tablas

1.1. Bandas de operación LTE por compañía en España	3
3.1. Temporización de las fases del proyecto	16
3.2. Planificación temporal de las tareas	16
3.3. Características Asus X554LD	17
3.4. Características Dell Vostro 270	18
3.5. Características Fujitsu Celsius M770X	18
3.6. Características Monitor ASUS VZ239HE	19
3.7. Características terminal móvil Samsung Galaxy A40	20
3.8. Características USRP National Instrument mo.2901	21
3.9. Características Antena Ettus VERT2450	21
3.10. Resumen coste recursos hardware	22
3.11. Resumen costes recursos humanos	24
3.12. Resumen presupuesto total	25
3.13. Resumen presupuesto total tras la modificación	27
6.1. Datos obtenidos en la prueba de rendimiento de nuestra red LTE para eNB con BW 5MHz	75
6.2. Datos obtenidos en la prueba de rendimiento de nuestra red LTE para eNB con BW 10MHz	76
6.3. Datos obtenidos en la prueba de rendimiento de nuestra red LTE para eNB con BW 20MHz	77

Siglas

<i>1G</i>	Primera Generación Móvil
<i>2G</i>	Segunda Generación Móvil
<i>3G</i>	Tercera Generación Móvil
<i>3GPP</i>	Third Generation Partnership Project
<i>4G</i>	Cuarta Generación Móvil
<i>5G</i>	Quinta Generación Móvil
<i>aGW</i>	Access Gateway
<i>API</i>	Application Program Interface
<i>APN</i>	Access Point Name
<i>ARM</i>	Advanced RISC Machines
<i>ARQ</i>	Automatic Repeat Request
<i>AS</i>	Access Stratum
<i>ASN.1</i>	Abstract Syntax Notation One
<i>BBU</i>	Base Band Unit
<i>BCCH</i>	Broadcast Control Channel
<i>BCH</i>	Broadcast Channel
<i>BIOS</i>	Basic Input Output System
<i>BW</i>	Bandwidth
<i>C – RAN</i>	Cloud Radio Access Network
<i>CCCH</i>	Common Control Channel
<i>CI</i>	Continuous Integration

<i>CPU</i>	Central Processing Unit
<i>DCCH</i>	Dedicated Control Channel
<i>DL – SCH</i>	Downlink Shared Channel
<i>DTCH</i>	Delicated Traffic Channel
<i>E – UTRAN</i>	Evolved Universal Terrestrial Radio Access Network
<i>eNB</i>	eNodeB
<i>EPC</i>	Evolved Packet Core
<i>FDM</i>	Frequency-Division Multiplex
<i>GB</i>	GigaByte
<i>GTPU</i>	GPRS Tunnelling Protocol User Plane
<i>GUTI</i>	Globally Unique Temporary Identity
<i>HDD</i>	Hard Disk Drive
<i>HSS</i>	Home Subscriber Server
<i>I – CSPP</i>	Interrogating-Call State Control Function
<i>ICIC</i>	Inter-Cell Interference Coordination
<i>ICMP</i>	Internet Control Message Protocol
<i>IMEI</i>	International Mobile Equipment Identity
<i>IMS</i>	IP Multimedia Subsystem
<i>IMSI</i>	International Mobile Subscriber Identity
<i>IoT</i>	Internet-of-Things
<i>IP</i>	Internet Protocol
<i>ISI</i>	Inter-symbol Interference
<i>LTE</i>	Long-Term Evolution
<i>LTS</i>	Long-Term Support
<i>MAC</i>	Medium-Access Control
<i>MBMS</i>	Multimedia Broadcast/Multicast Service
<i>MBSFN</i>	Multicast-Broadcast Single Frequency Network

MCC Mobile Country Code
MCCH Multicast Control Channel
MCH Multicast Channel
MIB Master Information Block
MIMO Multiple-Input Multiple-Output
MME Mobility Management Entity
MNC Mobile Network Code
MTCH Multicast Traffic Channel
NAS Non-Access Stratum
NAT Network Address Translation
NB – IoT Narrowband-IoT
NFAPI Network Functional API
NFV Network Function Virtualization
OAI Open Air Interface
OASIM Open Air Interface System Emulation
OFDM Orthogonal Frequency-Division Multiplexing
OFDMA Orthogonal Frequency-Division Multiple Access
OMV Operadora Móvil Virtual
OSA OpenAirInterface Software Alliance
P – CSCF Proxy-Call Session Control Function
P – GW Packet-Data Network Gateway
PAR Peak-to-Average Ratio
PBCH Physical Broadcast Channel
PC Persnal Computer
PCCH Paging Control Channel
PCFICH Physical Control Format Indicator Channel
PCH Paging Channel

PCRF Policy and Charging Rules Function
PDCCH Physical Downlink Control Channel
PDCCP Packet Data Convergence Protocol
PDSCH Physical Downlink Shared Channel
PHICH Physical Hybrid-ARQ Indicator Channel
PHY Physical Layer
PMCH Physical Multicast Channel
PNF Planar Near-field
PRACH Physical Random-Access Channel
PUCCH Physical Uplink Control Channel
PUSCH Physical Uplink Shared Channel
PXIe PCI eXtensions for Instrumentation
QoS Quality-of-Service
RAM Random Access Memory
RAN Radio Access Network
RCC Radio Control Center
RF Radio Frequency
RLC Radio-Link Control
ROCH Robust Header Compression
RRH Remote Radio Head
RRM Radio Resource Management
RRU Radio Remote Unit
S – CSCF Serving-Call Session Control Function
S – GW Serving Gateway
SC – FDMA Single Carrier - Frequency Division Multiple Access
SDN Software Defined Network
SDR Software Defined Radio

<i>SIM</i>	Subscriber Identity Module
<i>SPGW</i>	S-GW & P-GW
<i>SRS</i>	Sounding Reference Signal
<i>SSH</i>	Secure Shell
<i>TAC</i>	Technical Assistance Center
<i>TAI</i>	Tracking Area Identifier
<i>TCP</i>	Transmission Control Protocol
<i>TF</i>	Transport Format
<i>TFG</i>	Trabajo Fin de Grado
<i>TTI</i>	Transmission Time Interval
<i>UDP</i>	User Datagram Protocol
<i>UE</i>	User Equipment, the 3GPP name for the mobile terminal
<i>UGR</i>	Universidad de Granada
<i>UL – SCH</i>	Uplink Shared Channel
<i>UML</i>	Unified Modeling Language (Diagram)
<i>USIM</i>	Universal Subscriber Identity Module
<i>USRP</i>	Universal Software Radio Peripheral
<i>VM</i>	Virtual Machine
<i>VNF</i>	Virtualized Network Function
<i>WCDMA</i>	Wideband Code-Division Multiple Access
<i>Wi – Fi</i>	Wireless Fidelity
<i>WiMAX</i>	Worldwide Interoperability for Microwave Access
<i>XOR</i>	Exclusive OR (Operation)

Capítulo 1

Introducción

1.1. Contexto

Es impensable, a día de hoy, el desarrollo de nuestras vidas sin el uso diario de dispositivos móviles con acceso a la red. Cada vez tenemos sistemas de comunicaciones más complejos y para entenderlos es importante comprender de dónde vienen y cómo han ido evolucionando estos. El desarrollo de las tecnologías móviles comenzaron siendo una competencia nacional o regional pero, sin embargo, desde su fundación en diciembre de 1998, la iniciativa *Third Generation Partnership Project* (3GPP) mediante la participación de miles de personas ha convertido la tarea del desarrollo de las tecnologías móviles en una tarea desarrollada por una organización mundial.



Figura 1.1: Logo 3GPP [1]

Las tecnologías de comunicaciones móviles se dividen en generaciones, donde la primera generación (1G) eran sistemas de radio móvil analógicos en la década de los 80's. La segunda generación (2G) introdujo las comuni-

caciones móviles digitales y la tercera generación (3G) supuso el comienzo de los sistemas móviles de banda ancha.

El *Long Term Evolution* (LTE) es popularmente conocido como cuarta generación (4G), aunque muchos aseguran que realmente, el verdadero paso al 4G es realmente la versión 10 LTE, conocida como *LTE-Advanced*.

Más allá de los nombres y generaciones, lo importante son las capacidades reales del sistema y cómo han evolucionado, dando lugar a un enorme aumento de velocidad y de movilidad que se han ido consiguiendo con el paso de estas generaciones.

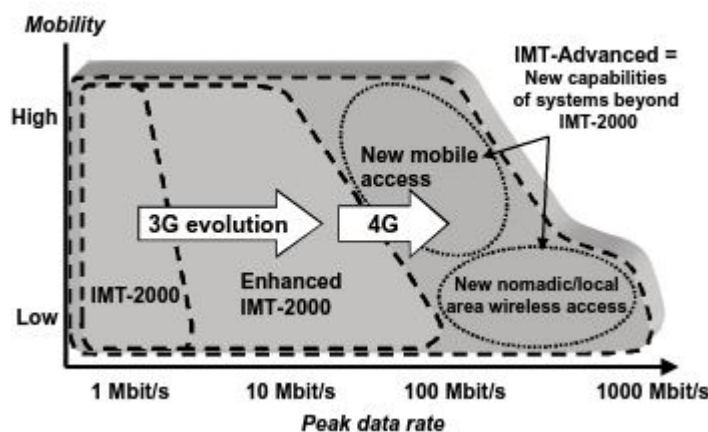


Figura 1.2: Tasa de datos máxima frente a movilidad en las diferentes generaciones [2]

1.1.1. LTE en España

En España contamos con muchas compañías telefónicas que ofertan un servicio de telefonía 4G. Sin embargo, muchas de ellas son Operadores Móviles Virtuales (OMVs) que no poseen infraestructura propia (se sugiere al lector remitirse a [10] donde encontrara todas las OMVs disponibles en España en febrero del año 2020). En España solo hay cuatro compañías de telefonía que cuentan con despliegue propio de infraestructura de red LTE, que son *Movistar*, *Vodafone*, *Orange* y *Yoigo*. Podemos ver en la tabla 1.1 las bandas de operación de cada una de las compañías anteriormente mencionadas basándonos en la referencia [11], y en la figura 1.3 según la referencia [12], podemos ver el porcentaje de cobertura de LTE por provincias en España en 2018.

Banda LTE	Frecuencia (MHZ)	Uso	Operadora
Banda 20	800	Después de ser liberada en 2015, es usado para 4G/LTE.	Movistar, Vodafone, Orange
Banda 32	1500	En desuso en la actualidad, a la espera de su subasta pública	-
Banda 3	1800	Es usada para 2G y 4G/LTE	Movistar, Vodafone, Orange, Yoigo
Banda 7	2600	Es usada para 4G/LTE	Movistar, Vodafone, Orange

Tabla 1.1: Bandas de operación LTE por compañía en España



Figura 1.3: Mapa porcentual de cobertura LTE por provincias en España

1.1.2. 5G en España

Durante el presente año 2020, se han comenzado en España del despliegue de redes de quinta generación (5G), el futuro próximo de las redes móviles. Mediante esta tecnología se conseguirá banda ancha de móvil de muy alta velocidad y capacidad, comunicaciones ultra fiables y de baja latencia (pasará a ser de 1 milisegundo frente a los 20-30 milisegundos de las redes LTE) y las comunicaciones masivas tipo máquina a máquina, donde se desarrollará el *Internet-of-Things* (IoT). Para profundizar información se remite al lector a la referencia [13].

1.2. Motivación

Este proyecto parte del deseo de visualizar de manera experimental la arquitectura, componentes y principales procedimientos de una red móvil mediante su configuración, puesta en marcha y análisis.

Para ello, el desarrollo de este proyecto se basa en una tecnología de red móvil que en la actualidad está completamente desplegada, como es el estándar de comunicaciones móviles 3GPP LTE.

1.3. Objetivos

El principal objetivo de este proyecto es la familiarización con una tecnología de red móvil que a día de hoy está completamente desplegada y da cobertura a casi la totalidad de la población a nivel mundial, concretamente se estima que un 95 % de la población está cubierta por la red 4G en los países desarrollados y un 70 % de la población en los países en vías de desarrollo. Para ello, se pretende implementar un escenario basado en las características del estándar LTE de 3GPP.

A causa de la adaptación de este Trabajo de Fin de Grado (TFG) sufrida por el plan de contingencia de la Universidad de Granada (UGR) provocado por la pandemia del Covid-19 [14] y la proclamación del estado de alarma en España del día 14 de marzo del 2020, esta sección estará dividida en dos subsecciones. La primera tratará los objetivos del escenario del proyecto inicial. La segunda tratará los del escenario proyecto adaptado.

1.3.1. Objetivos del escenario del proyecto inicial

Se pretende usar un conjunto de *Personal Computer* (PCs) que formen en su conjunto una red LTE. Uno de ellos estará conectado a un dispositivo *Software Defined Radio* (SDR), cuyo papel será funcionar como estación base eNB (*evolved Node B*). Como dispositivo de usuario UE (*User Equipment*) final se usará un terminal móvil comercial.

El primer PC funcionará como *Envolved Packet Core* (EPC) en cuyo interior estarán implementados las entidades *Mobility Management Entity* (MME), *Home Subscriber Server* (HSS), *Serving Gateway* (S-GW) y *Packet Data Network Gateway* (P-GW).

El PC conectado al SDR actuará como *Remote Radio Unit* (RRU) y otro PC, conectado entre la EPC y el RRU, actuará como *Radio Cloud Center* (RCC).

Este escenario será creado con el uso de la herramienta *Open Air Interface* (OAI) e implementará el uso de C-RAN (*Cloud Radio Access Network*), se remite al lector a la referencia [15].

Para finalizar, se realizará un análisis de las señales transmitidas por nuestra eNB y recibidas en nuestro terminal móvil (UE).

1.3.2. Objetivos del escenario del proyecto adaptado

Se pretende usar dos máquinas virtuales conectadas entre sí para la implementación de la red LTE. Una actuará como núcleo de red (EPC), donde se configurarán las entidades *Mobility Management Entity* (MME), *Home Subscriber Server* (HSS), *Serving Gateway* (S-GW) y *Packet Data Network Gateway* (P-GW). La otra máquina virtual actuará como emulador de eNB y de UE, gracias a la herramienta *Open Air Interface System Emulation* (OASIM).

Para finalizar, se realizará un análisis de rendimiento de nuestra red LTE creada con la herramienta *Open Air Interface* mediante el uso de inyecciones de tráfico con la herramienta *iperf*. Por último, se analizarán trazas *wireshark* para mostrar como se realiza la señalización.

1.4. Estructura de la memoria

En esta sección se enuncian y describen los diferentes capítulos de los que consta la presente memoria.

- **Capítulo 1. Introducción**

En este capítulo en primer lugar se proporciona información acerca del contexto en el que se va a trabajar. Acto seguido se indica la motivación de la realización del proyecto y por último se definen los objetivos que buscamos con la realización de éste.

- **Capítulo 2. Estado del arte.**

En este capítulo se exponen diferentes soluciones similares a la usada en el proyecto, *Open Air Interface*. También se explica por qué se ha decidido usar ésta.

- **Capítulo 3. Planificación y presupuesto.**

En este capítulo, en primer lugar, se definen las diferentes fases en las que se divide el desarrollo total del proyecto. En segundo lugar, se hace una estimación del coste asociado a cada tipo de recurso necesario para la realización del proyecto.

- **Capítulo 4. Fundamentos teóricos.**

En este capítulo se relatan todos los conceptos necesarios para la fundamentación teórica necesaria para la elaboración del proyecto.

- **Capítulo 5. Implementación.**

En este capítulo se redactan en detalle los pasos seguidos para la creación de nuestra red LTE con el uso de la herramienta *Open Air Interface*.

- **Capítulo 6. Evaluación de rendimiento.**

En este capítulo se realiza un análisis de rendimiento sobre la red creada y se mostrará la señalización de las trazas mediante *wireshark*.

- **Capítulo 7. Conclusiones.**

Este último capítulo culmina la memoria, donde se manifiestan las conclusiones a las que se ha llegado con la realización de este proyecto, así como una valoración personal y por último unas posibles líneas futuras.

- **Bibliografía.**

Incluye todas las fuentes usadas para la adquisición de los conocimientos necesarios para la realización del proyecto, así como para la elaboración de esta memoria.

- **Anexos.**

Incluye todos los archivos de configuración completos a los que se hace referencia en el capítulo 5 sobre la configuración de los distintos equipos.

Capítulo 2

Estado del arte

En este capítulo veremos otras posibles soluciones que ofrecen características similares a *Open Air Interface*. Al final, se explicará por qué finalmente se eligió OAI como herramienta para la elaboración del presente proyecto. Para este capítulo se hace uso de las referencias [16], [17], [18] y [8].

2.1. srsLTE

El *software srsLTE* es un *software* desarrollado por la compañía irlandesa *Software Radio Systems* (SRS), fundada en 2012, especializada en *software* de alto rendimiento para sistemas inalámbricos.

Se trata de una librería de código abierto escrita en lenguaje C para aplicaciones usando dispositivos SDR. Esta herramienta permite implementar la parte de eNB y el UE, soportando las características de la versión 8 del estándar (LTE *Release* 8). Para cubrir las funciones de seguridad y la red troncal, utiliza se puede utilizar la plataforma *OpenLTE*.

Los paquetes que incluye este *software* son:

- **srsUE**

Aplicación completa para la emulación de un terminal UE y que soporta todas las capas de red, desde el nivel física hasta la capa de red.

- **srsENB**

Aplicación para la implementación de una estación base eNB.

2.2. OpenLTE

Herramienta de código abierto para aplicaciones SDR. Utiliza el lenguaje de C++ e implementa la especificación 10 de 3GPP LTE.

Una de sus principales funciones es su uso para la prueba y simulación de la funcionalidad de transmisión y recepción de enlace descendente usando FDD, permitiendo el uso de UE para realizar pruebas de rendimiento y funcionalidades de red. Su núcleo de red EPC implementa las funcionalidades de HSS y de MME.

2.3. Amari LTE 100

El *software Amari LTE 100* es un *software* cerrado y de pago desarrollado por la compañía francesa *Amari*, fundada en 2012, cuyo objetivo es ofrecer algunas soluciones asequibles y de alta calidad a la comunidad 4G para liberar la creatividad y, en última instancia, expandir las comunicaciones entre las personas.

Sus características principales son las siguientes:

■ Características de eNB

- Capa física (PHY).

Cumple con la versión 13 de LTE. Se encarga de la realización del control de potencia UE de circuito cerrado. Se puede agregar soporte de otros cabezales de radio con una biblioteca compartida externa.

- Capa de protocolo.

Es compatible con la versión 13 de LTE. Implementa las capas *Medium-Access Control* (MAC), RLC, *Packet Data Convergence Protocol* (PDCP) y RRC.

- NB-IoT.

Es compatible con NB-IoT versión 13, su objetivo es dar soporte de UE NB1 de un solo tono y multitono. Dispone de todos los modos de operación (en banda, banda de protección y autónomo). Se pueden usar varias celdas NB-IoT y LTE al mismo tiempo en el mismo eNB. Dispone de soporte de múltiples niveles de cobertura.

■ Características de EPC

- MME.

Implementa un MME con S-GW, P-GW y HSS incorporados. Admite varios eNBs con interfaz S1 estándar (protocolos S1AP y GTP-U). Ofrece soporte de tarjetas *Universal Subscriber Identity Module* (USIM) utilizando el algoritmo de autenticación *Exclusive OR* (XOR) o *Milenage*. Se encarga del manejo de procedimientos de UE: conexión, autenticación, configuración de seguridad, desconexión, actualización del área de seguimiento, acceso al servicio, establecimiento del portador de radio, paginación. Dispone de base de datos de usuario configurable. No se necesita HSS externo. *Application Program Interface* (API) remota utilizando *WebSocket* y monitor de línea de comando.

- Servidor IMS (*Internet Protocol Multimedia Subsystem*).

Implementa *Proxy-Call Session Control Function* (P-CSCF) con *Interrogating-Call State Control Function* (I-CSCF), *Serving-Call Session Control Function* (S-CSCF) y HSS incorporados. Ofrece soporte de tarjetas USIM utilizando el algoritmo de autenticación XOR o *Milenage*. Ofrece también soporte de llamadas de voz entre UEs. Contiene un base de datos de usuario configurable. No se necesita HSS externo.

- Puerta de enlace MBMS (*Multimedia Broadcast/Multicast Service*).

Lista configurable por el usuario de servicio y componentes de multidifusión.

■ Interfaz gráfica de usuario basada en web

Muestra registros de archivos. Muestra registros en tiempo real a través de *WebSocket*. Muestra estadísticas de la capa física en gráficos y permite la visualización de asignación de bloque de recursos.

2.4. Open Air Interface

El *software Open Air Interface* es un *software* de código abierto y libre basado en el estándar LTE de 3GPP. Proporciona dos módulos *software* distintos, uno para la implementación del núcleo de red (EPC) y otro para la red de acceso.

Las dos partes en las que se divide son:

- **Openair-CN**

Implementa los diferentes equipos que forman el núcleo de la red (EPC), estos son el MME, el HSS y el S/P-GW.

- **Openairinterface5G**

Implementa la red de acceso mediante la implementación de la estación base eNB, y también existe la posibilidad de simular varios terminales de usuario UE.

Además, permite flexibilidad a la hora del montaje del escenario, por ejemplo, nos implementar los diferentes componentes en un mismo PC, o en diferentes.

2.5. Conclusiones y solución elegida

Como podemos ver, *Amari LTE 100* es la solución más completa de todas, ya que al ser de pago, su desarrollo es mucho más minucioso y además cuenta con mayor soporte.

OpenLTE, a pesar de no tener coste añadido a nuestro proyecto, no nos permitirían el desarrollo completo de nuestro objetivo de la creación de una red LTE, ya que no nos permite la simulación de UE para la parte de red de acceso y lo necesitamos para la evaluación de rendimiento de nuestro escenario. Además, el EPC que dispone es muy básico.

srsLTE tampoco sería una solución óptima ya que con esta herramienta no se podría desarrollar la simulación del escenario que queremos al completo. Sólo ofrece la posibilidad de implementar la red de acceso (eNB y UE), no permite la implementación del EPC.

En último lugar, tenemos *Open Air Interface*, plataforma elegida ya que es libre, implementa tanto el EPC como la red de acceso al completo así como la posibilidad de montar diferentes tipos de escenarios.

Capítulo 3

Planificación y presupuesto

Durante este capítulo se describe la planificación del proyecto que se va a llevar a cabo. Para ello, disponemos de una planificación de las tareas a realizar y analizaremos los recursos *hardware*, *software* y humanos necesarios para la implementación del proyecto una estimación del coste real de éste en la realización en un ámbito laboral.

3.1. Planificación y organización de las fases del proyecto

En este apartado se detallan y ordenan en orden cronológico las diferentes etapas planificadas para el desarrollo del proyecto.

- **Estudio de la tecnología 4G LTE**

Para empezar con el proyecto, comenzamos buscando información para conocer y familiarizarnos con la tecnología que vamos a trabajar (4G LTE), así como la historia de ésta para saber cuáles son los objetivos de esta generación y cómo los consigue con el uso de diferentes componentes y protocolos.

- **Investigación del mercado**

En segundo lugar, debemos investigar el mercado para ver que posibles recursos *software* podríamos usar para el desarrollo de nuestro proyecto, estudiando posibles candidatos y comparándolos entre sí para elegir el más conveniente.

- **Estudio de OpenAirInterface**

En tercer lugar, como requisito de la descripción del proyecto, usaremos el *software Open Air Interface*. Por lo tanto, debemos estudiar la herramienta, su funcionamiento y, en concreto, el escenario que vamos a montar usándola, conociendo cuáles son las versiones de sistema operativo que necesitamos para su correcto funcionamiento, cuál es su estructura para ver cuántos equipos y con qué requisitos los necesitamos, así como saber qué archivos de configuración e instalación usaremos.

- **Preparación del escenario real**

Pasamos ahora a la preparación de los equipos físicos que necesitaremos para el desarrollo e implementación del proyecto, estudiada en la fase anterior, y, además, debemos conocer cómo los debemos conectar entre ellos en una red interna así como la conexión NAT (*Network Address Translation*) al exterior.

- **Instalación de software y preliminares**

Esta fase consiste en la configuración del sistema operativo necesario, así como configurar los requisitos de kernel, gestión de la energía, instalación de git para la descarga de paquetes necesarios, etcétera.

- **Instalación y configuración de Open Air Interface**

Cuando ya tengamos todos los equipos en disposición, correctamente conectados y configurados, procedemos a la instalación y posterior configuración de OAI. Para ello, instalaremos los paquetes necesarios, modificaremos los archivos de configuración, etcétera.

- **Testeo y análisis técnico**

Esta fase consiste en la comprobación de que todo está correctamente conectado, configurado y en funcionamiento, para posteriormente visualizar los resultados.

- **Escritura de la memoria técnica**

Durante el desarrollo del proyecto se recopila toda la documentación necesaria, la información del proceso de implementación del proyecto, así como cualquier otra información relevante para el desarrollo de éste. Y, por último, se le da el formato requerido para la presentación.

Para observar de manera más gráfica y estructurada la organización temporal del desarrollo de las diferentes partes del proyecto antes del comienzo de éste, podemos tomar como referencia el diagrama de *Gantt* de la figura 3.1.

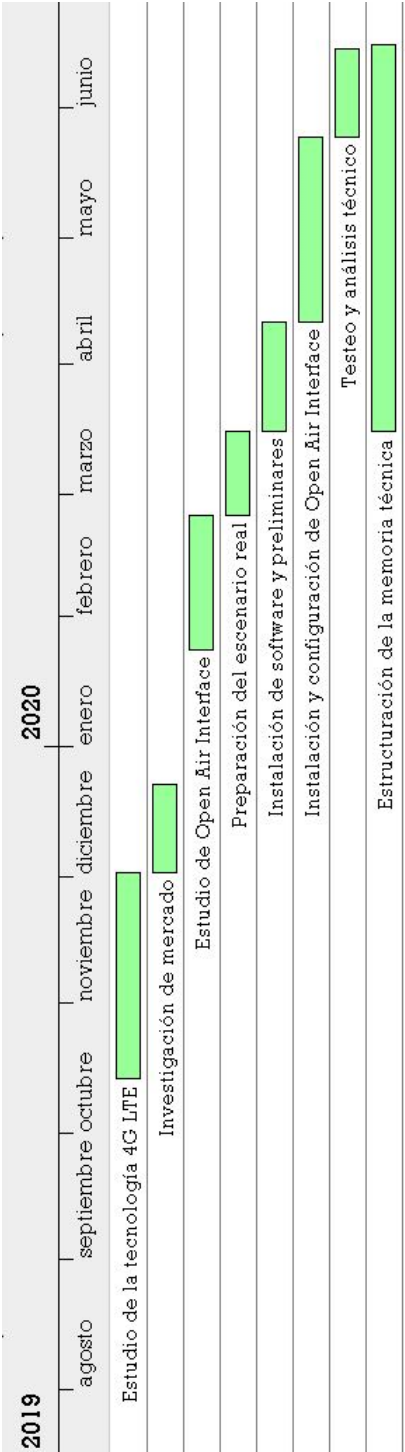



Figura 3.1: Diagrama de Grantt del proyecto

A continuación, se muestra en mayor detalle el inicio y fin de las diferentes etapas de realización del proyecto, así como una tabla en la que se indican la planificación de las horas planificadas de dedicación a cada una de etapas anteriores, en la tabla 3.1.



Nombre	Fecha de inicio	Fecha de fin
• Estudio de la tecnología 4G LTE	14/10/19	1/12/19
• Investigación de mercado	2/12/19	22/12/19
• Estudio de Open Air Interface	24/01/20	24/02/20
• Preparación del escenario real	25/02/20	15/03/20
• Instalación de software y preeliminares	16/03/20	10/04/20
• Instalación y configuración de Open Air Interface	11/04/20	24/05/20
• Testeo y análisis técnico	25/05/20	14/06/20
• Escritura de la memoria técnica	16/03/20	30/06/20

Tabla 3.1: Temporización de las fases del proyecto

Podemos ver ahora el desglose en horas de cada una de las fases de elaboración del proyecto en la tabla 3.2.

Etapas del proyecto	Horas
Estudio de la tecnología 4G LTE	40
Investigación de mercado	20
Estudio de Open Air Interface	45
Preparación del escenario real	25
Instalación de software y preliminares	15
Instalación y configuración de Open Air Interface	50
Testeo y análisis técnico	45
Escritura de la memoria técnica	80
TIEMPO TOTAL	320

Tabla 3.2: Planificación temporal de las tareas

3.2. Inventario y presupuesto

En este apartado se indican todos los recursos necesarios para el desarrollo del proyecto, los cuales los podemos dividir según su naturaleza en *hardware*, *software* y humanos. Al final, a modo de resumen, se pueden ver los costes estimados totales en la tabla 3.12.

3.2.1. Recursos hardware

Necesitaremos los siguientes recursos *hardware* para el desarrollo del proceso. Para esta subsección se hace uso de las referencias [19], [20], [21], [22], [23], [24] y [25].

■ Ordenador portátil Asus X554LD

Este equipo *hardware* será para uso personal, búsqueda de información y desarrollo de la memoria técnica. Sus características técnicas se muestran en la tabla 3.3.

Marca y modelo	Asus X554LD
Procesador	Intel Core i7-4510U Dual core (2.0 GHz, hasta 3.1 GHz (L3) Cache)
Memoria RAM	8 GB
Almacenamiento	1TB 5400 rpm SATA
Sistema Operativo	Windows 10
Tarjeta gráfica	NVIDIA GeForce GT 820M 1GB DDR3 VRAM

Tabla 3.3: Características Asus X554LD



Figura 3.2: Asus X554LD

■ Ordenador sobremesa Dell Vostro 270

Usaremos 2 ordenadores de este tipo para las partes que necesitan menos procesamiento del proyecto, es decir, para el RCC y el EPC. Sus características técnicas se encuentran en la tabla 3.4.

Marca y modelo	Dell Vostro 270
Procesador	Intel Core i3-3240, 3.4 GHz
Memoria RAM	4 GB
Almacenamiento	500 GB
Sistema Operativo	Ubuntu 18.04

Tabla 3.4: Características Dell Vostro 270



Figura 3.3: Dell Vostro 270

■ Ordenador sobremesa Fujitsu Celsius M770X

Usaremos un ordenador de este tipo para la parte del escenario que necesita más potencia de procesamiento, concretamente para el RRU. Las características principales se encuentran resumidas en la tabla 3.5.

Marca y modelo	Fujitsu Celsius M770X
Procesador	Intel Core i7-7820X, 3.6 GHz
Memoria RAM	32 GB
Almacenamiento	2.5 TB
Sistema Operativo	Ubuntu 18.04

Tabla 3.5: Características Fujitsu Celsius M770X



Figura 3.4: Fujitsu Celsius M770X

- **Monitor ASUS VZ239HE**

Usaremos tres monitores de este tipo, uno para cada uno de los PCs mencionados anteriormente. Sus características están indicadas en la tabla 3.6.

Marca y modelo	ASUS VZ239HE
Tamaño de pantalla	23"
Tipo de pantalla	LCD

Tabla 3.6: Características Monitor ASUS VZ239HE



Figura 3.5: Monitor ASUS VZ239HE

■ Terminal móvil Samsung Galaxy A40

Usaremos este terminal móvil para las distintas pruebas de evaluación de rendimiento. Podemos ver sus características más relevantes en la tabla 3.7.

Marca y modelo	Samsung Galaxy A40
Procesador	Exynos Samsung 7885 octacore (2x 1.8 GHz, 6x 1.6 GHz)
GPU	Mali-G71 MP2
Memoria RAM	4 GB
Almacenamiento	64 GB
Sistema Operativo	Android 10
Redes	2G (GSM 850/900/1800/1900) 3G (850/900/1900/2100) 4G (LTE 1(2100), 3(1800), 5(850), 7(2600), 8(900), 20(800), 38(2600), 40(2300), 41(2500))

Tabla 3.7: Características terminal móvil Samsung Galaxy A40



Figura 3.6: Terminal móvil Samsung Galaxy A40

■ USRP National Instrument mo. 2901

Usaremos este *Universal Software Radio Peripheral* (USRP) como SDR para implementar la estación base eNB, sus principales características podemos verlas en la tabla 3.8.

Fabricante y modelo	National Instrument mo. 2901
Razón máxima de tranferencia	15MS/s
Precisión en frecuencia	2.5ppm
Máximo ancho de banda	56MHz
Figura de ruido	5dB a 7dB

Tabla 3.8: Características USRP National Instrument mo.2901



Figura 3.7: USRP National Instrument mo.2901

▪ Antena Ettus VERT2450

Usaremos esta antena para emitir con el SDR mencionado anteriormente. Sus características las podemos observar en la tabla 3.9.

Fabricante y modelo	Ettus VERT 2450
Tipo	Vertical omnidimensional
Banda de emisión	Doble banda (de 2.4 a 2.48 GHz y de 4.9 a 5.9 GHz)
Ganancia	3dBi

Tabla 3.9: Características Antena Ettus VERT2450



Figura 3.8: Antena Ettus VERT2450

■ Tarjetas USIM

Usaremos la tarjeta USIM programable para poder acceder a nuestra red generada por la estación base. Lo haremos con el uso de tarjetas de la marca sysmocom y en concreto el producto sysmoUSIM-SJS1 SIM (*Subscriber Identity Module*) + USIM Card.

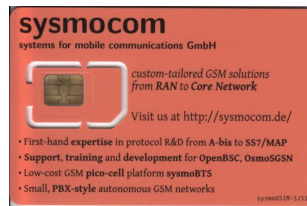


Figura 3.9: sysmoUSIM-SJS1 SIM + USIM Card

A modo de resumen, podemos ver en la tabla 3.10 el desglose de los costes de los diferentes recursos *hardware* necesarios.

Dispositivo Hardware	Número de unidades	Precio / unidad (€)	Precio total / tipo hardware (€)
Ordenador portátil	1	622,00	172,78 (10 meses)
Dell Vostro 270	2	360,00	720,00
Fujitsu Celsius M770X	1	2500,00	2500,00
Monitor	3	150,00	450,00
Terminal móvil	1	220,00	220,00
USRP	1	1636,00	1636,00
Antena	1	43,00	43,00
Tarjeta USI	1	8,00	8,00
		TOTAL	5749,78

Tabla 3.10: Resumen coste recursos hardware

3.2.2. Recursos software

Necesitaremos los siguientes recursos *software* para el desarrollo del proyecto. Usaremos las referencias [26], [27], [28], [29], [28], [30], [31], [32]

- **Sistema Operativo Ubuntu 18.04**

Sistema operativo de código abierto que usaremos en su versión más actual LTS (*Long Term Support*) para el montaje de nuestro escenario de trabajo.

- **Sistema Operativo Windows 10**

Sistema operativo preinstalado en el ordenador portátil de uso personal.

- **OAI-CN**

Esta herramienta es parte del paquete *Open Air Interface*. Mediante esta herramienta se crea la red troncal, es decir, el EPC y los diferentes componentes que lo componen.

- **Openairinteface5G**

Esta herramienta es parte del paquete *Open Air Interface*. Mediante esta herramienta se crea la estación base eNB.

- **MySQL**

Es un sistema para la gestión de bases de datos de carácter relacional desarrollado bajo la licencia dual (Licencia pública general y Licencia comercial) por *Oracle Corporation*. La usamos para gestionar la información de las tarjetas USIM.

- **GranttProject**

Herramienta de código abierto desarrollado para la gestión de proyectos. La hemos usado para la elaboración del diagrama de *Grantt* del proyecto.

- **Overleaf**

Herramienta de código abierto, colaborativa, basada en *LaTeX* en la nube. Se ha usado para la redacción del documento y para el seguimiento de esta por parte del tutor.

■ Lucidchart

Herramienta web gratuita que permite crear diagramas de flujo, organigramas, esquemas de sitios web, diseños *Unified Modeling Language* (UML), mapas mentales, prototipos de *software* y muchos otros tipos de diagrama. Usada para la realización de esquemas para su incorporación en la memoria técnica.

■ Wireshark

Herramienta gratuita usada para analizar los mensajes en una red de comunicación, donde se pueden observar los protocolos utilizados, los datos que se transmiten, etcétera. Usada para el análisis de la señalización de conexión entre nuestras distintas máquinas virtuales.

El coste de los recursos *software* estará íntegramente asociado al de la obtención de la licencia del sistema operativo *Windows 10*, siendo este de unos 150 €. El resto de los demás recursos son de código abierto por lo que no supondrán coste adicional.

3.2.3. Recursos humanos

En este apartado se incluyen las horas de trabajo invertidas por parte del estudiante, así como por parte del tutor para la realización del proyecto.

El número de horas de trabajo invertidas por el estudiante se estimó en el apartado 3.1 y las horas del tutor se estiman mediante el número de horas de tutorías, preparación del material, revisión, etc.

El precio por hora del tutor y del estudiante se estiman según su titulación. El precio por hora estimado para el tutor, Profesor Titular de la Universidad, es de 50 euros la hora y el precio por hora estimado para el estudiante, Graduado en Ingeniería de Tecnologías de Telecomunicación, es de 25 euros.

Podemos ver un desglose del total en la tabla 3.11.

Persona	Coste/h (€)	Tiempo (h)	Coste total / persona (€)
Alumno	25,00	320	8000,00
Tutor	50,00	20	1000,00
TOTAL			9000,00

Tabla 3.11: Resumen costes recursos humanos

3.2.4. Presupuesto total

A modo de resumen del capítulo, se ha realizado la suma para saber cuál sería el precio estimado del proyecto al completo incluyendo todos los recursos necesarios. El desglose del presupuesto por tipos de recurso está indicado en la tabla 3.12.

Tipo de recurso	Coste (€)
Hardware	5749,78
Software	150,00
Humano	9000,00
COSTE TOTAL	14899,78

Tabla 3.12: Resumen presupuesto total

3.3. Modificación de la planificación, inventario y presupuesto.

En esta sección se detallan las modificaciones debido a la reorganización sufrida por el plan de contingencia de la UGR provocada la pandemia del Covid-19 y la proclamación del estado de alarma en España el día 14 de marzo.

3.3.1. Modificación de la planificación

La planificación temporal apenas ha sufrido cambios. Justo se debió adaptar este TFG cuando ya está terminada la fase “Preparación del escenario real” por lo que dentro de la fase “Instalación del software y preliminares” se montará el nuevo escenario, en este caso de carácter virtual.

3.3.2. Modificación del inventario y presupuesto

La mayoría de los costes del inventario están asociados al material de preparación del escenario, por lo tanto, con el nuevo escenario a tratar el presupuesto se verá muy reducido.

26 3.3. Modificación de la planificación, inventario y presupuesto.

Nuevos recursos hardware

Como recursos *hardware*, solamente necesitaremos el ordenador portátil, que se usará tanto para uso personal como para la realización al completo del nuevo proyecto. No tendremos ningún recurso adicional a los mencionados anteriormente.

Nuevos recursos software

Como recursos *software*, mantendremos los siguientes del apartado 3.2.2. y serán usados para las mismas competencias del proyecto:

- Sistema Operativo Windows 10.
- OAI-CN.
- Openairinterface5G.
- MySQL.
- GranttProyect.
- Overleaf.
- Lucidchart.

Además, nos serán necesarios los siguientes.

- **Sistema Operativo Ubuntu 14.04 LTS**

Sistema operativo de código abierto usado para la creación de las máquinas virtuales donde montaremos nuestro escenario.

- **Oracle VM VirtualBox**

Software de virtualización de código abierto de la compañía *Oracle* con soporte para la creación de máquinas virtualizadas con una gran cantidad de sistemas operativos diferentes. Lo usaremos para la creación de dos máquinas virtuales con el sistema operativo *Ubuntu* 14.04 LTS para la realización de nuestro proyecto. [33]

- **Iperf**

Herramienta de *software* libre cuya función es realizar pruebas en redes informáticas. Su uso más común es la creación flujos de datos TCP (*Transmission Control Protocol*) y UDP (*User Datagram Protocol*) y

para medir el rendimiento de la red. Lo usaremos para inyectar tráfico en nuestra red para comprobar así el rendimiento de ésta. [34]

■ Microsoft Excel

Es una hoja de cálculo desarrollada por la compañía *Microsoft* para *Windows*, *macOS*, *Android* e *iOS*. Sus principales funciones son cálculo, herramientas gráficas y tablas. Está incluido dentro del paquete *software Microsoft Office*. Será usado en este proyecto para la creación de las gráficas del capítulo 6. [35]

De los nuevos recursos *software* sólo tiene un coste monetario la herramienta *Microsoft Excel* el cual es parte del paquete *Microsoft Office*. En concreto, usaremos el paquete llamado “*Microsoft 365 Personal*” cuyo coste es de 10€/mes, por lo que nos supondrá un coste total de 50€, ya que lo usaremos durante 5 meses. Los demás son de código abierto por lo que no nos supondrán un incremento en el coste del proyecto.

Nuevos recursos humanos

Los recursos humanos necesarios estimados para la implementación del nuevo proyecto son los mismos a los recursos de este tipo estimados en el apartado 3.2.3

Presupuesto total del nuevo proyecto

Una vez comprobadas todas las modificaciones en los diferentes tipos de recursos, podemos ver en la tabla 3.13 el nuevo desglose de costes del proyecto.

Tipo de recurso	Coste (€)
Hardware	172,78
Software	200,00
Humano	9000,00
COSTE TOTAL	9372,78

Tabla 3.13: Resumen presupuesto total tras la modificación

Capítulo 4

Fundamentos Teóricos

En este capítulo se describen todos los conceptos básicos necesarios para implantar los cimientos teóricos que fundamenten nuestro proyecto. Comenzaremos describiendo y conociendo la arquitectura, los componentes básicos y los principales procedimientos de la tecnología LTE. En segundo lugar, se explica de manera general *Open Air Interface*, el *software* en el que se basa nuestro proyecto. Veremos quién creó este *software*, cuáles son los objetivos que persigue y todos los componentes que está incluidos en el *software*. Por último, se describirá la arquitectura en la que se basa este tipo de *software* de emulación.

4.1. LTE

En esta sección conoceremos el fundamento teórico del estándar en el que basaremos nuestro proyecto, el estándar 3GPP LTE. El objetivo que presenta este estándar de comunicaciones móviles frente a su predecesor es el de conseguir velocidades de acceso a la red más elevadas que las obtenidas mediante el uso de la tecnología *Wideband Code-Division Multiple Access* (WCDMA) en la generación anterior. Se remite al lector a la referencias [2] para profundizar más en el estándar LTE.

En primer lugar, conoceremos cómo es su estructura y acto seguido profundizaremos un poco más conociendo sus principales procedimientos. Se remite al lector, si desea profundizar, se le recomienda la lectura de las referencias [2] y [36].

4.1.1. Estructura de la red LTE.

La arquitectura de la red LTE está muy claramente diferenciada en dos partes. Por un lado encontramos la red troncal, también conocida como EPC (*Evolved Packet Core*) y por otro lado encontramos la red de acceso, conocida como E-UTRAN (*Evolved Universal Terrestrial Radio Access Network*). Podemos ver en la figura 4.1 su esquema.

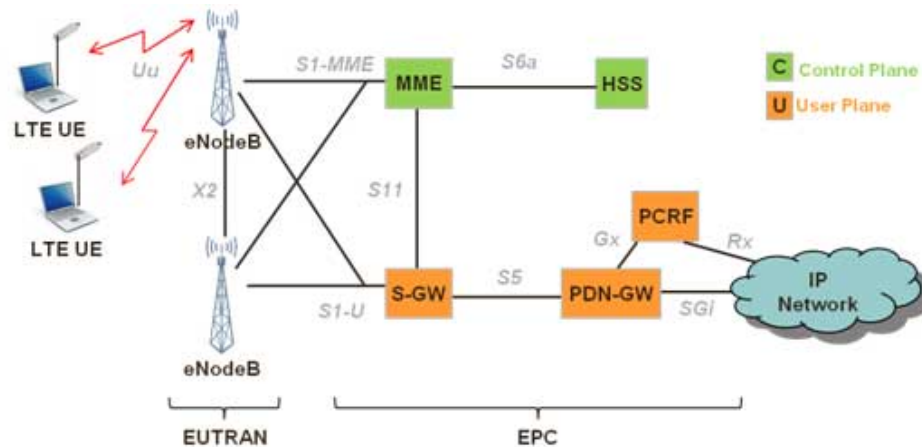


Figura 4.1: Arquitectura de red LTE [3]

Red troncal (EPC)

El EPC de LTE supone una evolución radical respecto a los núcleos de red de estándares predecesores. El EPC sólo soportaba inicialmente el acceso al dominio de conmutación de paquetes, sin acceso al dominio de conmutación de circuitos. Cuenta en su estructura con varios tipos de nodos diferentes, los cuales describimos a continuación. Podemos verlo en la figura 4.2.

■ MME

Este es el nodo del plano de control del EPC. Sus responsabilidades son la conexión/liberación de portadoras a un terminal, el manejo de las transiciones IDLE a ACTIVE y la gestión de las claves de seguridad necesarias.

La funcionalidad de operación entre el EPC y el terminal se conoce a veces como el estrato sin acceso (NAS, *Non-Access Stratum*), para separarlo del *Access Stratum* (AS) que gestiona la funcionalidad de operación entre el terminal y la red de acceso radio.

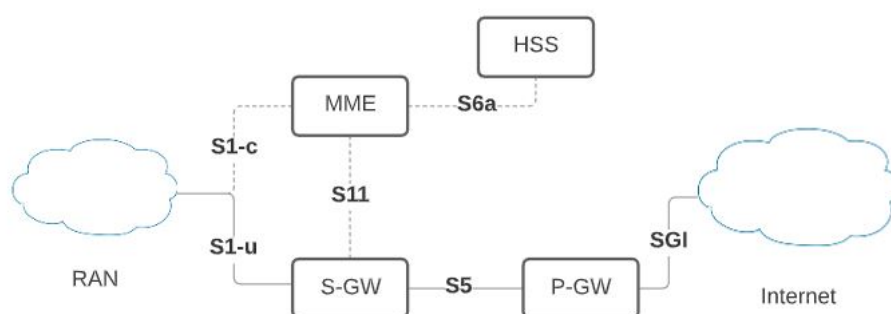


Figura 4.2: Arquitectura de EPC

■ HSS

Este nodo incluye una base de datos que contiene información de los subscriptores de la red.

■ S-GW

Este es el nodo de plano de usuario que conecta el EPC con el *Random Access Memory* (RAN) de LTE. El S-GW actúa como un ancla de movilidad cuando los terminales se mueven entre los diferentes eNBs, así como un anclaje de movilidad para otras tecnologías 3GPP (GSM/GPRS y HSPA).

La recolección de información y estadísticas necesarias para la tarificación de usuario también es manejada por el S-GW.

■ P-GW

Este nodo se encarga de la conexión del EPC a Internet. La asignación de la dirección IP (*Internet Protocol*) para un terminal específico es manejada por el P-GW, así como la aplicación de la calidad del servicio de acuerdo con la directiva controlada por el *Policy and Charging Rules Function* (PCRF).

El P-GW es también el ancla de movilidad para tecnologías de acceso radioeléctrico no 3GPP, como CDMA 2000, conectadas al EPC.

■ Elementos adicionales

• PCRF (Policy and Charging Rules Function)

Este nodo es el responsable de la gestión de calidad de servicio (*Quality-of-Service*, QoS) y tarificación. Tiene la capacidad para gestionar la política de redes y subscriptores en tiempo real,

la capacidad de enrutar y priorizar el tráfico de red de manera eficiente y dinámica. Tiene una vista unificada del contexto del suscriptor basada en una combinación de datos de dispositivo, red, ubicación y facturación.

- **MBMS (Multimedia Broadcast Multicast Service)**

Este nodo permite transmitir el mismo contenido a varios usuarios ubicados en un área específica, conocida como el área de servicio de MBMS (suele incluir varias celdas diferentes).

En cada celda que participa en la transmisión, se configura un recurso radio punto a multipunto y todos los usuarios que se suscriben al servicio MBMS reciben simultáneamente la misma señal transmitida.

Red de acceso (E-UTRAN)

En el estándar LTE de 3GPP, la red de acceso, también conocida como E-UTRAN, consiste en estaciones base o eNBs. Estas proporcionan las terminaciones del plano de usuario (*U-plane*) y plano de control (*C-plane*) hacia el móvil (UE).

Los eNBs están interconectados entre sí mediante la interfaz X2. Se supone que siempre existe una interfaz X2 entre los eNB que necesitan comunicarse entre sí. Los eNBs también están conectados por medio de la interfaz S1 al EPC. La interfaz S1 admite una relación de varios a varios entre a GWs y eNBs.

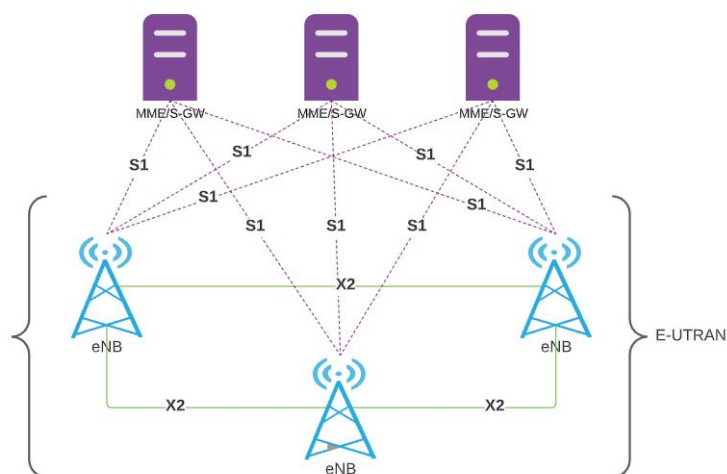


Figura 4.3: Arquitectura de E-UTRAN

E-UTRAN solo tiene un tipo de nodo, y este es el eNB.

■ eNB

El eNB es responsable de todas las funciones relacionadas con la radio en una o varias celdas. Es importante tener en cuenta que un eNB es un nodo lógico y no una implementación física.

Una implementación común de un eNB es un sitio de tres sectores, donde una estación base está manejando transmisiones de tres celdas. Aunque también se pueden encontrar otras implementaciones, como una unidad de procesamiento de banda base (BBU, *Base Band Unit*) a la que están conectadas varios cabezales de radio remotos (RRH, *Remote Radio Head*). Un ejemplo de esto último es un gran número de celdas interiores, o varias celdas a lo largo de una carretera, que pertenecen al mismo eNB.



Figura 4.4: Aspecto físico eNB (fabricante Khomp) [4]

Por lo tanto, una estación base es una posible implementación de un eNB. El eNB está conectado con el EPC por medio de la interfaz S1, más específicamente al S-GW por medio de la parte del plano de usuario S1, S1-u, y al MME por medio de la parte del plano de control S1, S1-c. Un eNB se puede conectar a varios MMEs/S-GW con el fin de compartir la carga y redundancia.

La interfaz X2, que conecta eNBs entre sí, se utiliza principalmente para admitir la movilidad en modo activo. Esta interfaz también se puede utilizar para las funciones de administración de recursos radioeléctricos (*Radio Resource Management*, RRM) multicelular, como

la coordinación de interferencias entre células, *Inter-Cell Interference Coordination* (ICIC). La interfaz X2 también se utiliza para soportar la movilidad sin pérdidas entre las celdas vecinas por medio del reenvío de paquetes.

4.1.2. Arquitectura de protocolos

La arquitectura de protocolos se puede dividir, por un lado, en la parte de usuario o plano de datos y, por otro lado, en el plano de control. Como se ve en la figura, muchas de las entidades son comunes a ambos planos. El plano de usuario se usa para el intercambio de mensajes entre dispositivos finales de usuario y el plano de control se usa para la señalización en la red. Podemos ver la pila de protocolos implicados en ambos planos en la figura 4.5.

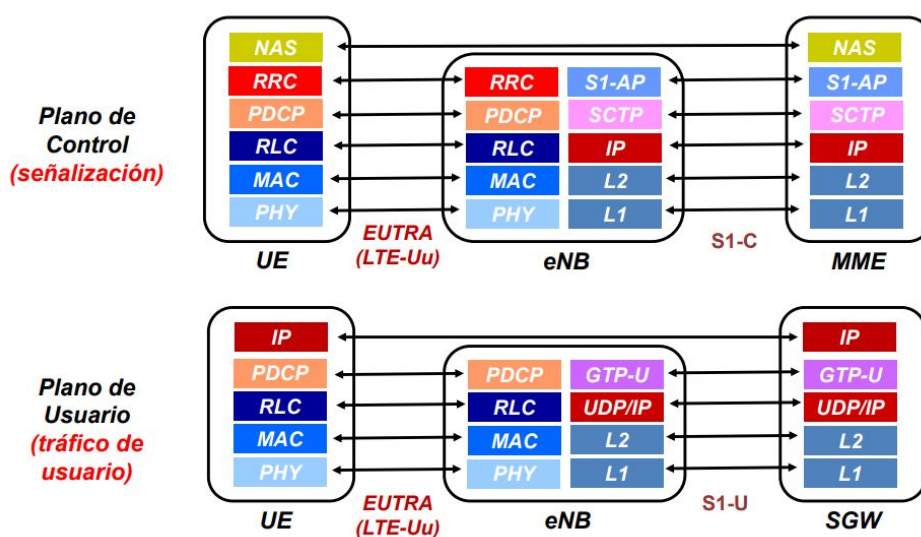


Figura 4.5: Pila protocolos plano de control y de usuario [5]

Los diferentes protocolos que forman parte de ambos planos son los siguientes:

■ PDCP

Packet Data Convergence Protocol (PDCP) realiza la compresión de encabezado IP para reducir el número de bits para transmitir a través de la interfaz radioeléctrica. El mecanismo de compresión de encabezado se basa en la compresión de encabezado robusta (*Robust Header*

Compression, ROHC), un algoritmo de compresión de encabezado estandarizado que también se utiliza para varias tecnologías de comunicación móvil.

PDCP también es responsable del cifrado y, para el plano de control, la protección de la integridad de los datos transmitidos, así como la entrega en secuencia y la eliminación de duplicados para el traspaso. En el lado del receptor, el protocolo PDCP realiza las operaciones de descifrado y descompresión correspondientes.

■ RLC

Radio-Link Control (RLC) es responsable de la segmentación/concatenación (ver esquema de realización en la figura 4.6), el manejo de las retransmisiones, la detección de duplicados y la entrega en secuencia a capas superiores.

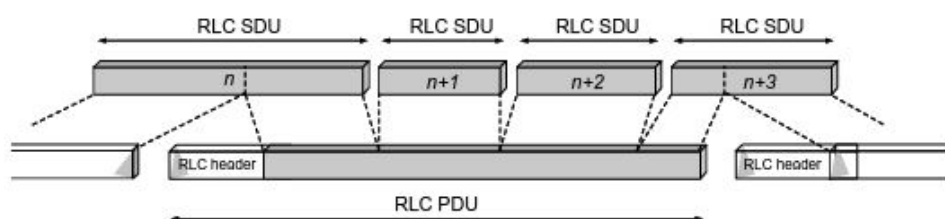


Figura 4.6: Segmentación/compresión RLC [2]

RLC proporciona servicios al PDCP en forma de portadoras radio.

■ MAC

Medium-Access Control (MAC) maneja la multiplexación de canales lógicos, las retransmisiones híbridas-ARQ (*Automatic Repeat reQuest*) y la planificación de los enlaces ascendente y descendente. La funcionalidad de planificación se encuentra en el eNB. La parte del protocolo híbrido-ARQ está presente en los extremos de transmisión y recepción del protocolo MAC.

MAC proporciona los servicios a RLC en forma de canales lógicos.

■ PHY

Physical Layer (PHY) maneja la codificación y decodificación, la modulación y demodulación, el mapeo multi-antena y otras funciones típicas de la capa física.

Esta ofrece servicios a la capa MAC en forma de canales de transporte.

4.1.3. Canales MAC

Para poder realizar su cometido mencionado anteriormente en la subsección 4.1.2, cuenta con tres tipos de canales: canales lógicos, canales de transporte y canales físicos.

Canales lógicos

A través de canales lógicos MAC proporciona los servicios al RLC .

Un canal lógico se define según el tipo de información que lleva y generalmente se clasifican:

- Como un canal de control, si se utiliza para la transmisión de la información de control y configuración necesario para el funcionamiento del sistema LTE.
- Como un canal de tráfico, si se utiliza para transmisión de datos de usuario.

Los diferentes tipos de canales lógicos especificados en el estándar LTE son los siguientes:

- **BCCH**

Broadcast Control Channel (BCCH), este tipo de canal es usado para la transmisión de información del sistema desde la red a todos los terminales de una célula. Antes de acceder al sistema, un terminal necesita adquirir la información del sistema para averiguar cómo se configura el sistema y, en general, cómo comportarse correctamente dentro de una celda.

- **PCCH**

Paging Control Channel (PCCH), este tipo de canal es usado para la paginación de terminales cuya ubicación en un nivel de celda no es conocida por la red, por lo tanto, el mensaje de paginación debe transmitirse en varias celdas.

- **CCCH**

Common Control Channel (CCCH), este tipo de canal es usado para la transmisión de información de control junto con el acceso aleatorio.

■ DCCH

Dedicated Control Channel (DCCH), este tipo de canal es usado para la transmisión de información de control hacia/desde un terminal, se utiliza para la configuración individual de terminales tales como diferentes mensajes de traspaso.

■ MCCH

Multicast Control Channel (MCCH), este tipo de canal es usado para la transmisión de información de control necesaria para la recepción del canal *Multicast Traffic Channel* (MTCH)

■ DTCH

Dedicated Traffic Channel (DTCH), este tipo de canal es usado para la transmisión de información de control hacia/desde un terminal, Este tipo de canal lógico es el utilizado para la transmisión de todos los datos de usuario de enlace ascendente y de enlace descendente.

■ MTCH

Multicast Traffic Channel (MTCH), este tipo de canal es usado para la transmisión de enlace descendente de MBMS.

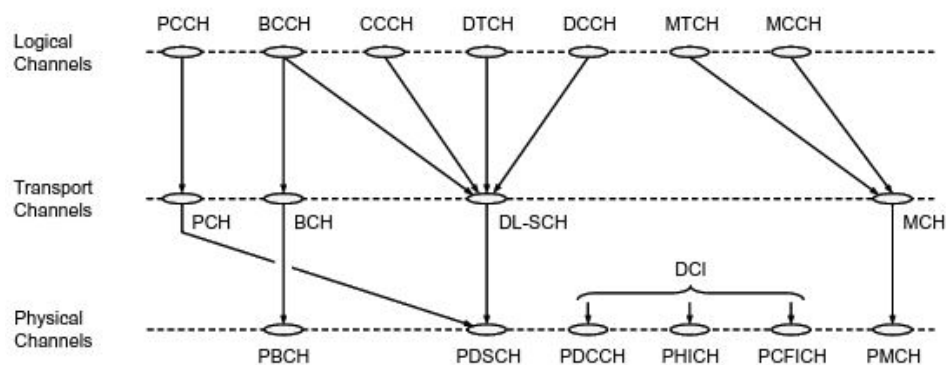


Figura 4.7: Mapa de canales en enlace descendente [2]

Canales de transporte

La capa MAC utiliza los servicios de la capa física (PHY) en forma de canales de transporte. Un canal de transporte se caracteriza por cómo y con qué características se transmite la información a través de la interfaz radioeléctrica. Los datos de un canal de transporte se crean en bloques de transporte y, en cada intervalo de tiempo de transmisión (*Transmission Time Interval*, TTI), como máximo se transmite un bloque de transporte de tamaño dinámico a través de la interfaz radioeléctrica hacia ó desde un terminal en ausencia de multiplexación espacial.

En el caso de la multiplexación espacial (MIMO), puede haber hasta dos bloques de transporte por cada TTI. Asociado con cada bloque de transporte hay un formato de transporte (*Transport Format*, TF), que especifica cómo se transmitirá el bloque de transporte a través de la interfaz radioeléctrica. El formato de transporte incluye información sobre el tamaño del bloque de transporte, su esquema de modulación y codificación así como asignación de antena. Al modificar el formato de transporte, la capa MAC puede conseguir distintas velocidades de datos. Por lo tanto, el control de velocidad también se conoce como selección de formato de transporte.

Los diferentes tipos de canales de transporte especificados en el estándar LTE son los siguientes:

- **BCH**

Broadcast Channel (BCH), tiene un formato de transporte fijo proporcionado por las especificaciones. Este tipo de canal se utiliza para la transmisión de partes de la información del sistema BCCH, concretamente el llamado Bloque de Información Maestra MIB (*Master Information Block*).

- **PCH**

Paging Channel (PCH), este canal es usado para la transmisión de información de paginación desde el canal lógico PCCH. El PCH admite la recepción discontinua (DRX) para permitir que el terminal ahorre energía de la batería, despertando sólo para recibir el PCH en los instantes de tiempo que estén predefinidos.

- **DL-SCH**

Downlink Shared Channel (DL-SCH), este tipo de canal es el principal canal de transporte utilizado para la transmisión de datos de enlace descendente en LTE. Es compatible con características LTE clave como la adaptación de velocidad dinámica y la programación dependiente

del canal en los dominios de tiempo y frecuencia, ARQ híbrido con combinación suave y multiplexación espacial, también es compatible con DRX para reducir el consumo de energía del terminal mientras que todavía proporciona una experiencia siempre activa.

El DL-SCH también se utiliza para la transmisión de las partes de la información del sistema BCCH no asignadas al BCH. Puede haber varios DL-SCH en una celda, uno por terminal programado en este TTI y, en algunos subtramas, un DL-SCH que lleva información del sistema.

■ MCH

Multicast Channel (MCH), este tipo de canal se utiliza para admitir MBMS.

Se caracteriza por un formato de transporte semiestático y una programación semiestática, en el caso de la transmisión multicelda mediante *Multicast-Broadcast Single Frequency Network* (MBSFN), la configuración del formato de programación y transporte se coordina entre los puntos de transmisión implicados en la transmisión MBSFN.

■ UL-SCH

Uplink Shared Channel (UL-SCH), este tipo de canal es el equivalente a DL-SCH pero de enlace ascendente, es decir, es el canal usado para el transporte de enlace ascendente utilizado para la transmisión de datos de enlace ascendente.

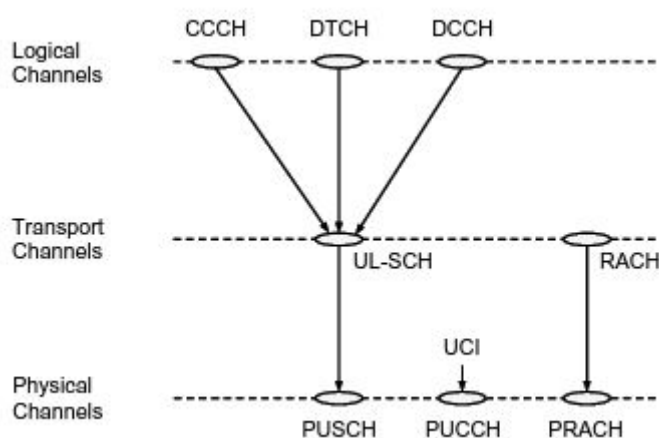


Figura 4.8: Mapa de canales en enlace ascendente [2]

4.1.4. Canales físicos

Un canal físico corresponde al conjunto de recursos de frecuencia de tiempo utilizados para la transmisión de un canal de transporte determinado y cada canal de transporte se asigna a un canal físico correspondiente.

Los tipos de canales físicos especificados en el estándar LTE son:

- **PDSCH**

Physical Downlink Shared Channel (PDSCH), este es el principal tipo de canal físico utilizado para la transmisión de datos de *unicast*, pero también es usado para la transmisión de información de paginación.

- **PBCH**

Physical Broadcast Channel (PBCH), este tipo de canal lleva parte de la información del sistema, requerida por el terminal para acceder a la red.

- **PMCH**

Physical Multicast Channel (PMCH), este tipo de canal es usado para la operación MBSFN.

- **PDCCH**

Physical Downlink Control Channel (PDCCH), este tipo de canal se utiliza para la información de control de enlace descendente, principalmente las decisiones de programación, necesarias para la recepción de PDSCH, y para la programación de subvenciones que permiten la transmisión en el PUSCH.

- **PHICH**

Physical Hybrid-ARQ Indicator Channel (PHICH), este tipo de canal lleva el acuse de recibo híbrido-ARQ para indicar al terminal si un bloque de transporte debe ser retransmitido o no.

- **PCFICH**

Physical Control Format Indicator Channel (PCFICH), este tipo de canal proporciona a los terminales la información necesaria para decodificar el conjunto de PDCCH. Sólo hay un PCFICH por portador de componentes.

- **PUSCH**

Physical Uplink Shared Channel (PUSCH), este tipo de canal es el equivalente, pero de enlace ascendente, al PDSCH. Hay como máximo un PUSCH por portadora de componentes de enlace ascendente por terminal.

- **PUCCH**

Physical Uplink Control Channel (PUCCH), este tipo de canal es utilizado por el terminal para enviar confirmaciones de ARQ híbridos, indicando al eNB si los bloques de transporte de enlace descendente se recibieron correctamente o no, para enviar informes de estado de canal que ayudan a la programación dependiente del canal de enlace descendente y para solicitar recursos para transmitir datos de enlace ascendente. Hay como máximo un PUCCH por terminal.

- **PRACH**

Physical Random-Access Channel (PRACH), este tipo de canal es usado para acceso aleatorio.

4.1.5. Tecnologías usadas en LTE

OFDM

OFDM es la tecnología que se usa en el enlace descendente ya que, mediante el uso de esta, se mejora la eficiencia espectral, se reduce el efecto ISI (*Inter-symbol Interference*) *multipath* y se proporciona mejor protección contra el *fading*. Puede ser extensivo a *Orthogonal Frequency-Division Multiple Access* (OFDMA), donde a cada usuario se le asigna un conjunto diferente de portadoras.

La tecnología OFDM puede ser vista como un tipo de transmisión multiportadora. Las características básicas de la transmisión OFDM son las siguientes:

- OFDM utiliza un número muy grande de subportadoras de banda relativamente estrecha. Por el contrario, una extensión multiportadora directa normalmente consistiría en sólo unas pocas subportadoras, cada una con un ancho de banda relativamente amplio. Un ejemplo lo podemos ver en la evolución multiportadora HSPA, la cual consta de un ancho de banda de transmisión global de 20 MHz y está formada por cuatro subportadoras, cada una con un ancho de banda del orden

de 5 MHz. Sin embargo, la transmisión OFDM puede constar de varios cientos de subportadoras transmitidas a través del mismo enlace de radio a un mismo receptor.

- El dominio en frecuencias de sus subportadoras está ajustado con un espaciado de subportadoras de $1/T$, donde T es el tiempo de símbolo de modulación por subportadora. Por lo tanto, podemos decir que el espaciado de la subportadora es igual a la velocidad de modulación por subportadora $1/T$.

Se puede observar en la figura 4.9 de manera más gráfica, como sería el espectro de una señal OFDM.

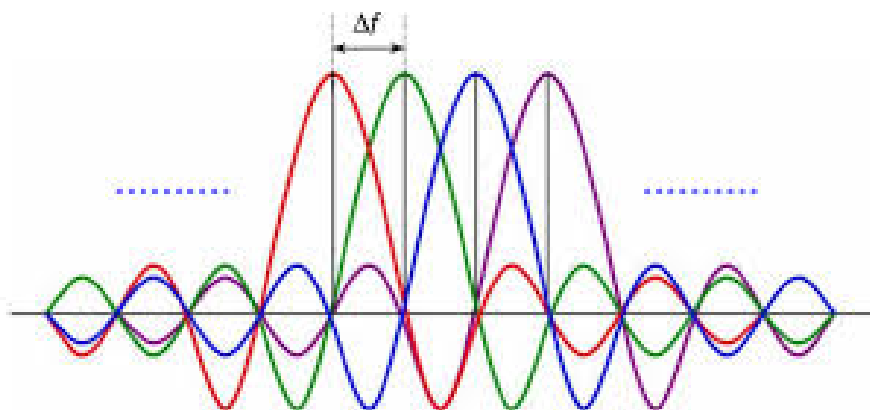


Figura 4.9: Espectro señal OFDM con cuatro canales [6]

SC-FDMA

Single Carrier Frequency Division Multiple Access (SC-FDMA) es la tecnología que se usa en el enlaces ascendentes ya que tiene una estructura y rendimiento similares a OFDMA, pero su ventaja es el beneficio en ahorro de batería y en eficiencia energética del que se benefician los terminales móviles, ya que tiene una menor *Peak-to-Average Ratio* (PAR).

La principal diferencia entre la transmisión OFDMA y SC-FDMA es que en enlace ascendente, cada subportadora transporta información acerca de cada símbolo de modulación, mientras que en el enlace descendente, cada subportadora sólo transporta información relacionada con un símbolo específico de modulación.

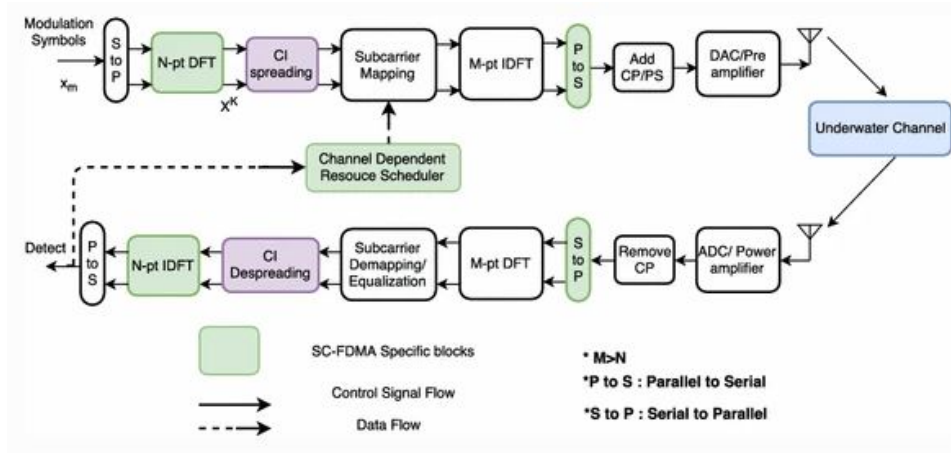


Figura 4.10: Proceso SC-FDMA [7]

MIMO

Multiple-input Multiple-output se basa en el uso de múltiples señales simultáneas de un mismo canal que aprovecha la propagación multicamino para incrementar la eficiencia espectral mediante el uso de diversas antenas, estrictas técnicas y complejos algoritmos de tratamiento de señal.

Para equipos monoantena, la tecnología multicamino era un gran inconveniente. Sin embargo, MIMO hace de este inconveniente su principal ventaja para multiplicar capacidad, y esto sirve para incrementar la velocidad, el caudal efectivo, el rango, la capacidad y la fiabilidad del sistema. Además, lo consigue sin incremento del ancho de banda ni de la potencia transmitida. Para profundizar en esta tecnología se remite al lector a la referencia [37].

4.2. Open Air Interface

La misión de *OpenAirInterface Software Alliance* (OSA) es proporcionar *software* y herramientas para la investigación inalámbrica 4G, 5G y el desarrollo de productos.

4.2.1. Objetivos perseguidos con su creación de código abierto

La generación actual de *hardware/software* para la red de acceso radioeléctrico (RAN) consiste en un gran número de elementos propietarios que



Figura 4.11: Logo Open Air Interface [8]

sofocan la innovación y aumentan el costo para que los operadores implementen nuevos servicios/aplicaciones en redes celulares de ritmo rápido en constante cambio.

El *software* de código abierto que se ejecuta en procesadores de uso general (como x86, ARM (*Advanced RISC Machines*)) puede simplificar en gran medida el acceso a la red, reducir los costos, aumentar la flexibilidad, mejorar la velocidad de innovación y acelerar el tiempo de comercialización para la introducción de nuevos servicios.

Ya hay un movimiento en la industria sobre el desarrollo de conceptos de redes definidas por *software* (*Software Defined Network*, SDN) para abrir las interfaces propietarias para controlar el *hardware/software* RAN.

Al mismo tiempo, el código abierto ha hecho un impacto muy significativo en las extremidades de las redes actuales, a saber, en los terminales debido al ecosistema *Android* y en la infraestructura en la nube debido, en parte, al ecosistema *OpenStack*. Estiman que una implementación de código abierto de pila en tiempo real (eNB, UE y red principal) en procesadores de propósito general cuando se combina con SDN, virtualización de funciones de red (*Network Function Virtualization*, NFV) y *OpenStack*, aporta una eficiencia significativa en el diseño RAN tanto desde la perspectiva de innovación como de costos.

4.2.2. OpenAirInterface Software Alliance

OSA es una organización francesa sin ánimo de lucro, creada en 2014 y cuya financiación proviene de patrocinadores corporativos. Su consejo está compuesto por representantes de los miembros estratégicos/asociados de la alianza.

OSA proporciona un marco establecido para la propiedad intelectual y las contribuciones financieras para contribuir al desarrollo de *software* y, al mismo tiempo, limitar la exposición legal potencial para nuestros comprometidos con el proyecto. La alianza también promueve el proceso meritocrático en el que los miembros individuales pueden contribuir al desarrollo de *software* OSA del *software* principal o los proyectos dirigidos por varias corporaciones miembros/organizaciones sin ánimo de lucro.

La alianza también se encarga de la organización de eventos, *training* y conferencias sobre el *software* OAI. La alianza también tiene la intención de proporcionar una colaboración abierta entre los miembros de su comunidad para fomentar la innovación y trabajar en el actual diseño de redes inalámbricas 4G y en el futuro diseño de redes inalámbricas 5G.

OSA actualmente proporciona una implementación compatible con el estándar de un subconjunto de la versión 10 LTE para UE, eNB, MME, HSS, SGW y PGW en equipos informáticos basados en Linux estándar (arquitecturas *Intel x86 PC/ARM*). El *software* es distribuido con carácter libre por la Alianza bajo los términos estipulados por el modelo de licencia OSA. Se puede utilizar junto con equipos de laboratorio de *Radio Frequency* (RF) estándar disponibles en muchos laboratorios (es decir, plataformas como *National Instruments/Ettus USRP* y *PCI eXtensions for Instrumentation* (PXIe), además de *hardware* RF personalizado proporcionado por EURECOM para implementar estas funciones en un grado suficiente para permitir la interoperación en tiempo real con dispositivos comerciales.

Algunos usuarios industriales ya han estado trabajando en sistemas basados en OAI integrados con equipos de radiofrecuencia móvil desplegables comercialmente y han proporcionado demostraciones en las principales ferias industriales (*Mobile World Congress Asia 2014*, *Mobile World Congress Barcelona* en 2013, *Industrial Maintenance Innovation Conference* 2013).

El objetivo principal es proporcionar una implementación de referencia de código abierto que siga el proceso de normalización 3GPP a partir de Rel-13 y el camino evolutivo hacia el 5G y que esté disponible libremente para la experimentación en equipos de laboratorio de productos básicos.

4.2.3. EURECOM

EURECOM es el miembro fundador de la OSA. Es una escuela de posgrado francesa y un centro de investigación en sistemas de comunicación, seguridad digital y ciencia de datos.

Tiene su sede en el parque científico internacional de *Sophia Antipolis*



Figura 4.12: Logo EURECOM [9]

dentro del nuevo Campus *SophiaTech*, que reúne a universidades de renombre como TELECOM *ParisTech* y otras universidades europeas como la Universidad Politécnica de Turín, la Universidad de *Aalto* (anteriormente Universidad Tecnológica de *Helsinki*), la Universidad Técnica de *Múnich* y la Universidad Noruega de Ciencia y Tecnología.

4.2.4. Componentes del software OAI

El *software OpenAirInterface* está compuesto por los siguientes directorios:

- **ci-scripts**

Meta-scripts utilizados por el proceso OSA *Continuous Integration* (CI). Contiene también archivos de configuración utilizados día a día por CI.

- **cmake_targets**

Usado para construir utilidades para compilar (simulación, emulación y plataformas en tiempo real) y archivos de compilación generados.

- **common**

Contiene algunas utilidades comunes de OAI. Otras herramientas se pueden encontrar en *openair2/UTILS*.

- **doc**

Contiene una lista de conjuntos de características actualizadas.

- **LICENSE**

Contiene la licencia del *software*.

- **maketags**

Script para generar etiquetas de código en emacs.

- **nfapi**

Contiene el código *Network Functional API* (NFAPI) que define un protocolo de red que se utiliza para conectar una función de red física (*Planar Near-field*, PNF) que ejecuta LTE capa 1 a una función de red virtual (*Virtualized Network Function*, VNF) que ejecuta LTE capa 2 y superior.

- **openair1**

Este directorio contiene las funciones de codificación/decodificación del canal de transporte para *OpenAirInterface* y un subconjunto de LTE/*Worldwide Interoperability for Microwave Access* (WiMAX)/ *Wi-Fi* (*Wireless Fidelity*) *MODEMS*. Solo se pueden compilar en plataformas x86.

- **openair2**

Este directorio contiene los *scripts* y *Makefiles* necesarios para generar todas las estructuras de datos de configuración LTE basadas en el código fuente *Abstract Syntax Notation One* (ASN.1) RRC de las especificaciones 36.331 RRC. Estas estructuras son utilizadas por MAC y PHY para fines de configuración.

- **openair3**

Este directorio contiene el código para GPP LTE Rel10 para implementar el eNB y el UE, usando los protocolos S1AP, NAS GTPV1-U.

- **targets**

Código para los envoltorios de nivel superior para simulación unitaria para canales PHY, emulación a nivel de sistema (eNB-UE con y sin S1), y eNB y UE y RRH GW en tiempo real.

Capítulo 5

Implementación

En este capítulo se desarrolla la implementación del proyecto al completo, desde la creación de las máquinas virtuales para que sea posible la virtualización de nuestro escenario, hasta la puesta en funcionamiento de todos sus componentes detallando todo el proceso intermedio hasta llegar a tener nuestro escenario a pleno rendimiento.

5.1. Creación de las máquinas virtuales

El primer paso es la instalación local del software de virtualización para poder crear con él las máquinas virtuales, en este proyecto se usará el *software* mencionado en el apartado 3.3.2, es decir, *Oracle VM VirtualBox*. Se recomienda instalar la última versión así como su pack de extensión asociado.

Una vez entramos en disposición del *software* de virtualización, pasamos a la creación de las máquinas virtuales, las cuales tendrán las siguientes características para proporcionar un rendimiento eficiente a nuestro escenario:

- **Memoria RAM:** 2 *GigaBytes* (GB)
- **Memoria *Hard Disk Drive* (HDD):** 30 GB
- **Procesador(es):** 1 *Central Processing Unit* (CPU)
- **Sistema operativo:** Ubuntu 14.04 LTS

Una vez creadas dos máquinas virtuales, instalaremos como sistema operativo de ambas *Ubuntu* 14.04 LTS. Una de las máquinas se llamará EPC,

la cual contendrá la parte de red trocal de nuestro escenario. La otra se llamará OAISIM, donde simularemos la red E-UTRAN y un UE comercial.

Antes de pasar a la instalación de los diferentes paquetes *software* y la configuración de estos, debemos hacer los cimientos de nuestra red, es decir, definir su estructura lógica. Para ello, desde *Oracle VM VirtualBox* debemos acceder a los ajustes de red de las dos máquinas virtuales que hemos creado y añadir un segundo adaptador, que se utilizará para crear una red interna entre ambas máquinas.

Una vez disponemos del adaptador de red interna pasamos a la configuración de la red en ambas máquinas. El diseño de red que deseamos podemos verlo en la figura 5.1, donde vemos las diferentes interfaces de las máquinas virtuales, de conexión interna y de conexión externa. Para la conexión externa se usará una tarjeta de red del ordenador portátil que se pondrá en modo NAT. En la figura también aparecen las direcciones IPs fijas de las diferentes interfaces que configuraremos a continuación.

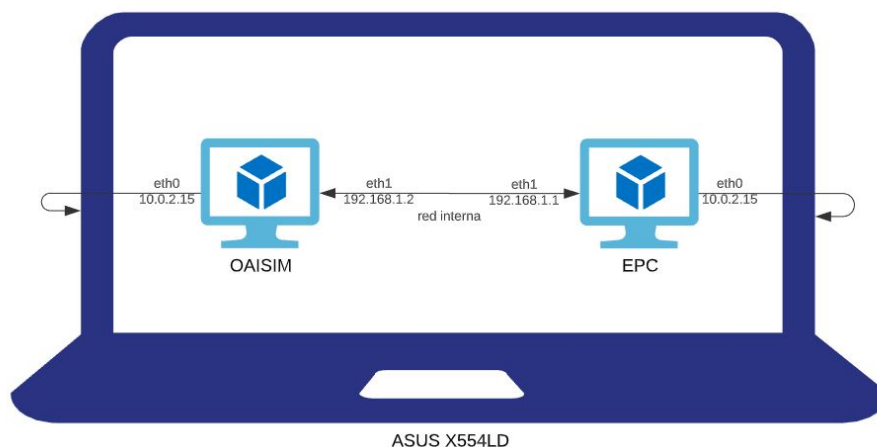


Figura 5.1: Estructura lógica de la red del escenario.

Como podemos ver, para evitar confusiones futuras, ambas máquinas usan sus respectivas interfaces eth1 para la conexión entre ellas en la red interna y sus respectivas eth0 para la conexión a la tarjeta de red del PC.

Establecimiento de las direcciones IPs fijas

Para establecer las direcciones IPs fijas que deseamos, debemos modificar el archivo `/etc/network/interfaces` incluyendo en las diferentes interfaces las direcciones que se establecen en el diseño lógico de la figura 5.1.

Una vez realizados y guardados los cambios en ambas máquinas, pasamos a reiniciar las interfaces de red (en ambas máquinas) para que se apliquen los cambios. Esto lo podemos hacer de dos maneras:

Podríamos hacerlo de manera directa mediante el comando:

```
$ sudo /etc/init.d/networking restart
```

También podríamos reiniciarlas de manera manual, deshabilitando y volviendo a habilitar las dos interfaces:

```
$ sudo ifconfig eth0 down  
$ sudo ifconfig eth0 up  
$ sudo ifconfig eth1 down  
$ sudo ifconfig eth1 up
```

Podemos comprobar que tienen conectividad entre sí y hacia Internet con el uso de *ping*.

```
manuel@ubuntu:~$ ping 192.168.1.1  
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.83 ms  
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.905 ms  
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.968 ms  
^C  
--- 192.168.1.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2003ms  
rtt min/avg/max/mdev = 0.905/1.236/1.836/0.425 ms  
manuel@ubuntu:~$ ping google.es  
PING google.es (172.217.17.3) 56(84) bytes of data.  
64 bytes from mad07s09-in-f3.1e100.net (172.217.17.3): icmp_seq=1 ttl=54 time=37  
.7 ms  
64 bytes from mad07s09-in-f3.1e100.net (172.217.17.3): icmp_seq=2 ttl=54 time=39  
.9 ms  
64 bytes from mad07s09-in-f3.1e100.net (172.217.17.3): icmp_seq=3 ttl=54 time=39  
.1 ms  
64 bytes from mad07s09-in-f3.1e100.net (172.217.17.3): icmp_seq=4 ttl=54 time=36  
.4 ms  
^C  
--- google.es ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3005ms  
rtt min/avg/max/mdev = 36.406/38.297/39.909/1.354 ms
```

Figura 5.2: Prueba de conexión OAISIM

```

manuel@epc:~$ ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.798 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.866 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.857 ms
^C
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.798/0.840/0.866/0.038 ms
manuel@epc:~$ ping google.es
PING google.es (172.217.16.227) 56(84) bytes of data.
64 bytes from mad08s04-in-f3.1e100.net (172.217.16.227): icmp_seq=1 ttl=55 time=
30.9 ms
64 bytes from mad08s04-in-f3.1e100.net (172.217.16.227): icmp_seq=2 ttl=55 time=
34.4 ms
64 bytes from mad08s04-in-f3.1e100.net (172.217.16.227): icmp_seq=3 ttl=55 time=
30.6 ms
64 bytes from mad08s04-in-f3.1e100.net (172.217.16.227): icmp_seq=4 ttl=55 time=
30.8 ms
^C
--- google.es ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 30.670/31.736/34.453/1.572 ms

```

Figura 5.3: Prueba de conexión EPC

Por último, a modo de resumen de las diferentes interfaces de red que contienen cada una de las máquinas virtuales, se pueden observar las figuras 5.4 y 5.5.

```

manuel@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:86:e0:65
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe86:e065/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:306 errors:0 dropped:0 overruns:0 frame:0
          TX packets:285 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:51722 (51.7 KB)  TX bytes:47249 (47.2 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:66:57:89
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe66:5789/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:61 errors:0 dropped:0 overruns:0 frame:0
          TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8865 (8.8 KB)  TX bytes:9251 (9.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:7176 (7.1 KB)  TX bytes:7176 (7.1 KB)

```

Figura 5.4: Interfaces de red OAISIM

```
manuel@epc:~$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:27:d6:e5
          Direc. inet:10.0.2.15  Difus.:10.0.2.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe27:d6e5/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
          Paquetes RX:11185 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:5093 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:8789920 (8.7 MB)  TX bytes:340737 (340.7 KB)

eth1      Link encap:Ethernet  direcciónHW 08:00:27:d8:1a:e0
          Direc. inet:192.168.1.1  Difus.:192.168.1.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fed8:1ae0/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
          Paquetes RX:28 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:73 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:2601 (2.6 KB)  TX bytes:9833 (9.8 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO MTU:65536 Métrica:1
          Paquetes RX:45 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:45 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1
          Bytes RX:4480 (4.4 KB)  TX bytes:4480 (4.4 KB)
```

Figura 5.5: Interfaces de red EPC

5.2. Instalación

5.2.1. Instalación de preliminares

Antes de comenzar a instalar los paquetes de OAI necesarios para la implementación, debemos realizar una serie de preliminares para que el software funcione correctamente. Los preliminares consisten, en primer lugar, en deshabilitar los estados C de la BIOS (*Basic Input Output System*). En segundo lugar, se desactiva el escalado de frecuencia de la CPU, para después instalar *Kernel* de baja frecuencia y por último instalar *git* para obtener los repositorios de OAI. Debemos hacerlo para nuestras dos máquinas virtuales.

■ Deshabilitado de los estados C de la BIOS

Procedemos en primer lugar a eliminar todas las funciones de administración de energía de la BIOS (estados de sueño, en particular C-states). Para conseguir esto debemos editar el archivo `/etc/default/grub`.

Para deshabilitar el estado p y el estado c en Linux, debemos agregar las siguientes líneas (la segunda es de carácter opcional):

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_idle.max_cstate=0 intel_pstate=disable" processor.max_cstate=1 idle=poll
```

Para que se haga efectivo el cambio debemos introducir el siguiente comando en la terminal:

```
$ sudo update-grub
```

Por último, en el archivo `/etc/modprobe.d/blacklist.conf` debemos añadir la siguiente línea al final de éste (en caso de que el archivo no exista, debemos de crearlo nosotros mismos):

```
blacklist intel_powerclamp
```

Para comprobar el estado de la CPU, instalaremos la herramienta *i7z*, donde para instalarla podemos hacerlo mediante el comando:

```
$ sudo apt-get install i7z
```

Y para lanzarla debemos usar el comando:

```
$ sudo i7z
```

Debemos comprobar que la CPU no cambie su frecuencia en más de 1-2 hercios y no debe haber ningún estado C que no sea C0. Es importante comprobar esto, ya que si la frecuencia de la CPU está cambiando o está en estado C, debemos comprobar el proceso anterior, ya que si esto no está correctamente configurado tendremos futuros problemas de tiempo real con el eNB/UE.

■ Desactivación del escalado de frecuencia de la CPU

Pasamos al escalado de frecuencia, para ello instalamos la herramienta `cpufrequtils`, para ello usaremos el comando:

```
$ sudo apt-get install cpufrequtils
```

Ahora debemos editar el archivo `/etc/default/cpufrequtils` (en caso de que no exista el archivo debemos crearlo). En este archivo debemos añadir la siguiente línea:

```
GOVERNOR="performance"
```

Una vez guardamos nuestro archivo debemos desactivar `ondemand` daemon, de lo contrario una vez que reiniciemos todos estos cambios desaparecerán, lo haremos mediante los siguientes comandos:

```
$ sudo update-rc.d ondemand disable $ /etc/init.d/cpufrequtils  
restart
```

Por último comprobamos que está todo bien configurado con el comando:

```
$ cpufreq-info
```

Una vez realizado esto, reiniciar ambas máquinas virtuales y comprobar de nuevo con el comando anterior para ver que no se han modificado los parámetros.

■ Instalación de Kernel de baja frecuencia

Instalamos el *kernel* de baja frecuencia en su versión 3.19, la cual es la versión de kernel asociada a nuestra versión de *Ubuntu* 14.04 LTS de 64-bit.

```
$ sudo apt-get install linux-image-3.19.0-61-lowlatency linux-headers-3.19.0-61-lowlatency
```

Tras introducir el comando para que se haga efectivo debemos de reiniciar y comprobar después con el siguiente comando que ha sido correctamente instalado, en la versión que queremos y en la fecha en la que se realizó su instalación.

```
$ uname -a
```

■ Instalación de git

Para descargar los repositorios necesarios posteriormente de *gitlab*, necesitamos instalar *git*, esto lo haremos mediante el comando:

```
$ sudo apt-get install git
```

5.2.2. Instalación de paquetes necesarios

En este apartado instalaremos todos los paquetes necesarios de OAI para la realización de nuestro proyecto tras los anteriores preliminares instalados en nuestras máquinas virtuales.

Por un lado, a una de nuestras máquinas virtuales, a la que llamamos OAISIM, le instalaremos el paquete *Openairinterface5g*, con el que conseguiremos la simulación del eNB y del UE. Por otro lado, a nuestra otra máquina virtual, llamada EPC, le instalaremos el paquete *openair-cn*, con el que conseguiremos la simulación del EPC, el cual estará formado por el MME, el HSS y el S/P-GW.

Instalación de OAISIM

En primer lugar, debemos descargar el repositorio de *gitlab* necesario para la instalación del paquete *openairinterface5g*.

Podemos descargarlo con el siguiente comando mediante el cual descargamos el repositorio sin necesidad de loguearnos en el servidor *gitlab* usando claves SSH (*Secure Shell*):

```
$ git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
```

Una vez descargado el repositorio procedemos a su instalación mediante el comando (sólo es necesario hacer este procedimiento una vez para que se instalen los paquetes):

```
$ sudo /openairinterface5g/cmake_targets/build_oai -I
```

Mediante el uso de la opción “-I”, se instalan todos los paquetes necesarios para la implementación del paquete “build_oai”.

Instalación de EPC

En primer lugar debemos descargar el repositorio de *gitlab* necesario para la instalación del paquete *openair-cn*.

Podemos descargarlo con el siguiente comando mediante el cual descargamos el repositorio. En este caso, sí que nos pedirá usuario y contraseña de *gitlab* y es necesario acceder previamente al sitio web de *gitlab* para crearse una cuenta.

```
$ git clone https://gitlab.eurecom.fr/oai/openair-cn.git
```

De manera adicional, también nos descargaremos un paquete de herramientas extra para OAI, lo haremos mediante el comando:

```
$ git clone https://gitlab.eurecom.fr/oai/xtables-addons-oai.git
```

Una vez descargado el repositorio y antes de pasar a la instalación de los diferentes paquetes, necesitamos especificar FQDM (Fully Qualified Domain Name) para el EPC. Para ello vamos a editar el archivo `/etc/hosts` incluyendo en este los siguientes detalles de configuración:

```
127.0.0.1 localhost
127.0.1.1 epc.5GLaboratory epc
127.0.1.1 hss.5GLaboratory hss
```

Donde “epc” es el *hostname*, y “5GLaboratory” es el dominio.

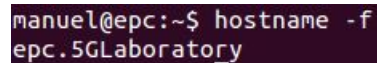
Una vez guardamos el archivo, es necesario reiniciar la máquina para que los cambios sean efectivos.

Una vez está reiniciada nuestra *Virtual Machine* (VM) EPC, podemos

comprobar que está correcta la nueva configuración mediante el comando

```
$ hostname -f
```

El resultado que obtendremos será “epc.5GLaboratory”, como podemos ver en la figura 5.6.



```
manuel@epc:~$ hostname -f  
epc.5GLaboratory
```

Figura 5.6: Hostname EPC

A continuación accedemos al directorio `/openair-cn/SCRIPTS` y usamos los siguientes comandos para instalar los tres componentes del EPC:

```
$ /build_mme -i  
$ /build_hss -i  
$ /build_spgw -i
```

Mediante el uso de la opción “-i”, se instala por primer vez el paquete que se indica en ese comando para la compilación de cada uno de los equipos que componen nuestro EPC, por lo que esto es solamente necesario ejecutarlo la primera vez.

5.3. Configuración

Al igual que en el apartado de instalación, podemos diferenciar en dos partes la configuración. Por un lado, configuraremos el OAISIM, y por otro el EPC, donde configuraremos de manera individual sus componentes MME, HSS y SPGW (S-GW y P-GW).

5.3.1. Configuración OAISIM

En este apartado configuraremos la máquina virtual OAISIM una vez ya lo instalamos en la sección anterior, para su posterior puesta en marcha.

Para su configuración, debemos modificar `enb.band7.tm1.usrpb210.conf`,

archivo alojado en el directorio `/openairinterface5g/targets/PROJECTS/GENERIC-LTE-EPC/CONF`.

Podemos ver en el anexo este archivo al completo. Sin embargo, los campos que debemos modificar son los siguientes, quedando los demás con sus valores por defecto:

- **tracking_area_code**

Código de área de seguimiento, debe coincidir en ambas partes de nuestra red LTE, es decir, debemos poner el mismo aquí que en el EPC.

Le daremos el valor “1”.

- **mobile_country_code**

Código móvil del país, al igual que en el campo anterior, debemos poner el mismo aquí que en el EPC.

Le daremos el valor “208”.

- **mobile_network_code**

Código de la red móvil, al igual que en los campos anteriores, debemos poner el mismo aquí que en el EPC.

Le daremos el valor “92”.

- **Parámetros MME**

- **mme_ip_address**

Este campo a su vez está dividido en otros cuatro subcampos.

- **ipv4**

Dirección IP versión 4 del mme.

Le daremos el valor “192.168.12.70”.

- **ipv6**

Dirección IP versión 6 del mme.

Le daremos el valor “192:168:30::17”.

- **active**

Yes/No, en función de si queremos que esté activo el mme.

Le daremos el valor “yes”.

- **preference**

Indica cual de las dos versiones (4 ó 6) preferimos de IP.

Le daremos el valor “ipv4”.

- **NETWORK INTERFACES**

Este campo a su vez esta dividido en otros cinco subcampos.

- **ENB_INTERFACE_NAME_FOR_S1_MME**

Nombre del interfaz de red s1 mediante el cual nos conectamos desde el eNB al MME.

Le daremos el valor “eth0”.

- **ENB_IPV4_ADDRESS_FOR_S1_MME**

Dirección IP versión 4 del interfaz de red s1 definido anteriormente.

Le daremos el valor “192.168.12.150/24”.

- **ENB_INTERFACE_NAME_FOR_S1U**

Nombre del interfaz de red s1 mediante el cual se enviarán los datos de usuario.

Le daremos el valor “eth0”.

- **ENB_INTERFACE_NAME_FOR_S1U**

Dirección IP versión 4 del interfaz de red s1 usado para el envío de datos de usuario definido anteriormente.

Le daremos el valor “192.168.12.150/24”.

- **ENB_PORT_FOR_S1U**

Número de puerto usado para la interfaz de envío de datos de usuario.

Le daremos el valor “2152”.

5.3.2. Configuración MME

En este apartado configuraremos el MME dentro de la máquina virtual EPC, una vez ya está instalado openair-cn en la sección anterior, para su posterior puesta en marcha.

En primer lugar, antes de pasar a la configuración del MME, debemos hacer un paso previo en la VM EPC, el cual es necesario para la instalación de los otros dos paquetes que forman el EPC además del EPC, por lo que este paso sólo será necesario realizarlo una vez.

Consiste en copiar los archivos de configuración en el directorio `/usr/local/etc/oai`, en el caso de que este directorio no exista debemos crearlo.

El proceso descrito anteriormente se realiza mediante la introducción de los siguientes comandos:

```
$ sudo mkdir -p /usr/local/etc/oai/freeDiameter
$ sudo cp /openair-cn/ETC/mme.conf /usr/local/etc/oai
$ sudo cp /openair-cn/ETC/hss.conf /usr/local/etc/oai
$ sudo cp /openair-cn/ETC/spgw.conf /usr/local/etc/oai
$ sudo cp /openair-cn/ETC/acl.conf /usr/local/etc/oai/freeDiameter
$ sudo cp /openair-cn/ETC/mme_fd.conf /usr/local/etc/oai/freeDiameter
$ sudo cp /openair-cn/ETC/hss_fd.conf /usr/local/etc/oai/freeDiameter
```

Ahora ya sí, estamos en disposición de poder configurar el MME, para ello, debemos modificar el archivo `/usr/local/etc/oai/mme.conf`, donde los campos que debemos modificar son los siguientes, quedando los demás con sus valores por defecto:

- **REALM**

Indica el nombre del dominio.

Le daremos el valor “5GLaboratory”.

- **s6A**

Está compuesto por dos campos:

- **S6A_CONF**

Indica la localización del archivo de configuración del interfaz S6A.

Le daremos el valor “`/usr/local/etc/oai/freeDiameter/mme_fd.conf`”.

- **HSS_HOSTNAME**

Indica el nombre del equipo HSS.

Le daremos el valor “hss”.

■ GUMMEI_LIST

Lista que contiene el identificador de entidad de administración móvil único a nivel mundial. Está compuesta por cuatro campos:

- **Mobile Country Code (MCC)**
Indica el código móvil del país.
Le daremos el valor “208”.
- **Mobile Network Code (MNC)**
Indica el código de la red móvil.
Le daremos el valor “93”.
- **MME_GID**
Indica el identificador de grupo de MME.
Le daremos el valor “4”.
- **MME_CODE**
Indica el código de MME.
Le daremos el valor “1”.

■ TAI_LIST

Tracking Area Identifier List (TAI_LIST), lista que contiene identidades de área de seguimiento. Incluye tres campos, MCC y MNC, que deben ser iguales que en “GUMMEI_LIST”, y el campo “TAC” (*Technical Assistance Center*), que es el código asociado de operador y tomará el valor “1”.

■ NETWORK_INTERFACES

Está compuesto por 5 campos, son los siguientes:

- **MME_INTERFACE_NAME_FOR_S1_MME**
Indica el nombre de la interfaz S1 que conecta con la máquina OAISIM.
Le daremos el valor “eth1”
- **MME_IPV4_ADDRESS_FOR_S1_MME**
Indica el la dirección IP de la tarjeta de red mediante la que conecta con la máquina OAISIM.
Le daremos el valor “192.168.1.1/24”.

- **MME_INTERFACE_NAME_FOR_S11_MME**

Indica el nombre de la interfaz S11 usada para la conexión con el S/P-GW. El salto será local ya que están dentro de la máquina EPC.

Le daremos el valor “lo”.

- **MME_IPV4_ADDRESS_FOR_S11_MME**

Indica la dirección IP de la interfaz S11 usada para la conexión con el S/P-GW.

Le daremos el valor “127.0.11.1/8”.

- **MME_PORT_FOR_S11_MME**

Indica el puerto para la conexión con la interfaz S11.

Le daremos el valor “2123”.

- **S-GW**

En su interior tiene el siguiente campo:

- **SGW_IPV4_ADDRESS_FOR_S11**

Indica la dirección IP del S-GW.

Le daremos el valor “127.0.11.2/8”.

Nota: el archivo completo se puede consultar en el anexo.

También, en referencia al MME, tenemos que modificar el archivo de configuración *freeDiameter* asociado a MME (/usr/local/etc/oai/freeDiameter/mme_fd.conf), el cual podemos encontrar completo en el anexo.

Los campos que debemos modificar son los siguientes, quedando los demás con sus valores por defecto:

- **Identity**

Indica la identidad dentro del dominio.

Le daremos el valor “epc.5GLaboratory”.

- **Realm**

Indica el nombre del dominio.

Le daremos el valor “5GLaboratory”.

■ ConnectPeer

Indica el nombre del otro equipo al que estará conectado, en este caso será con el HSS.

Le daremos el valor “hss.5GLaboratory”.

Además, este campo está compuesto por una lista, incluye los siguientes campos:

- ConnectTo

Indica la dirección IP del equipo HSS.

Le daremos el valor “127.0.0.1”.

- port

Indica el puerto de conexión con el equipo HSS.

Le daremos el valor “3868”.

- realm

Indica el dominio en el que está el equipo HSS.

Le daremos el valor “5GLaboratory”.

También debemos añadir en la lista lo siguiente. “No_SCTP”, para que no use SCTP, en caso de usarse deshabilitaría el uso del protocolo TCP y sólo se conectaría y escucharía en SCTP. “No_IPv6”, para forzar a que use la dirección IP versión 4 y no la versión 6. “Prefer_TCP” para que use TCP preferiblemente sobre SCTP. “No_TLS” para que no use protocolo TLS, ya que no es necesario que para la conexión dentro de nuestra máquina virtual se haga de manera segura, reduciendo así requisitos de procesamiento.

5.3.3. Configuración HSS

En este apartado configuraremos el HSS dentro de la máquina virtual EPC, una vez ya está instalado openair-cn en la sección anterior, para su posterior puesta en marcha.

El primer paso antes de configurar el archivo de configuración, será la instalación de la herramienta *MySQL*, la cual es una base de datos que usaremos para el HSS.

En primer lugar, usaremos el siguiente comando para instalarla.

```
$ sudo apt-get install mysql-server
```

Reiniciamos el servidor, para su puesta en funcionamiento, mediante el comando:

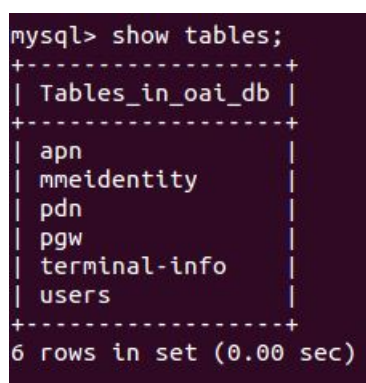
```
$ sudo /etc/init.d/mysql restart
```

Una vez está en funcionamiento *MySQL*, debemos importar la que usa nuestra herramienta OAI, la cual viene incluida en el paquete *Openair-CN*. Ésta es `/openair-cn/SRC/OAI_HSS/db/oai.db.sql`.

Para importar la base de datos mencionada anteriormente, lo haremos mediante el siguiente comando, donde “root” es nuestro usuario de *MySQL*, `oai_db` es el nombre de la base de datos y `oai.db.sql` es el archivo en formato base de datos de *MySQL* a importar:

```
$ mysql -u root -p oai_db <oai.db.sql
```

Una vez la tenemos importada podemos acceder a ella y vemos que dispone del listado de tablas que podemos ver en la figura 5.7.



```
mysql> show tables;
+-----+
| Tables_in_oai_db |
+-----+
| apn               |
| mmeidentity       |
| pdn               |
| pgw               |
| terminal-info     |
| users             |
+-----+
6 rows in set (0.00 sec)
```

Figura 5.7: Listado de tablas base de datos de OAI.

Dentro de la tabla “mmeidentity” añadimos nuestras dos entradas, es decir, `epc` (`epc.5GLaboratory`) y `hss` (`hss.5GLaboratory`). Acto seguido podemos comprobar entrando a esta, que se encuentran disponibles además las que vienen definidas por defecto. Podemos ver el interior de la tabla “mmeidentity” en la figura 5.8.

```
mysql> select * from mmeidentity;
```

idmmeidentity	mmehost	mmerealm	UE-Reachability
2	mme2.openair4G.eur	openair4G.eur	0
1	nano.openair4G.eur	openair4G.eur	0
5	abeille.openair4G.eur	openair4G.eur	0
4	yang.openair4G.eur	openair4G.eur	0
3	mme3.openair4G.eur	openair4G.eur	0
6	calisson.openair4G.eur	openair4G.eur	0
7	epc.5GLaboratory	5GLaboratory	0
8	hss.5GLaboratory	5GLaboratory	0

```
8 rows in set (0.00 sec)
```

Figura 5.8: Tabla mmeidentity de la base de datos de OAI.

Otras listas importantes en nuestra base de datos son las siguientes:

▪ pdn

En esta tabla podemos ver el APN y otros parámetros asignados a los *IP Multimedia Subsystem* (IMSI) de los usuarios. Los más importantes son el tipo de pdn, su pdn, el IMSI de cada identidad y sus umbrales de velocidad permitida tanto para enlace ascendente como enlace descendente.

▪ apn

En esta tabla podemos ver información sobre los APN (*Access Point Name*).

▪ users

En esta tabla encontramos la información de los usuarios de la red. Incluye información como el IMSI, número de teléfono, *International Mobile Equipment Identity* (IMEI), “mmeidentity_idmmeidentity”, *key* (clave de cifrado usada), “sqn” (usado para sincronizar la tarjeta SIM y la red para la autenticación), “rand” (puede configurarse o dejarse vacío), “OPc” (parámetro usado por el operador para grabar los datos en las tarjetas SIM).

El archivo de configuración que debemos modificar es el archivo `/usr/local/etc/oai/hss.conf`, que podemos encontrarlo completo en el anexo, donde los campos que debemos modificar son los siguientes, quedando los demás con sus valores por defecto:

- **MYSQL_server**

Indica la dirección IP del servidor de *MySQL*.

Le daremos el valor “127.0.0.1”.

- **MYSQL_user**

Indica el nombre de usuario de la base de datos de *MySQL*.

Le daremos el nombre por defecto “root”.

- **MYSQL_pass**

Indica la contraseña del usuario de la base de datos de *MySQL*.

Le daremos el valor “5GLaboratory”.

La contraseña recibe el mismo nombre que el dominio usado, pero no tienen conexión entre sí, se usa por facilidad de recordarla, ya que es la usada en varios procesos, pero podríamos haber usado cualquier otra.

- **MYSQL_db**

Indica el nombre de la base de datos de *MySQL*.

Le daremos el valor “root”.

También tenemos que modificar el archivo de configuración freeDiameter asociado a HSS (/usr/local/etc/oai/freeDiameter/hss_fd.conf), el cual podemos encontrar completo en el anexo.

En este caso solo tenemos que modificar dos campos, los cuales están ya definidos en el archivo de configuración de freeDiameter del MME. Por lo tanto, en esta sección sólo se indican los valores que debemos darle. Los demás campos permanecerán con sus valores por defecto:

- **Identity**

Le daremos el valor “hss.5GLaboratory”.

- **Realm**

Tomará el valor “5GLaboratory”.

5.3.4. Configuración SPGW

En este apartado configuraremos el SPGW, el cual está de la máquina virtual EPC, una vez ya está instalado *openair-cn* en la sección anterior, para su posterior puesta en marcha.

En este caso, para configurar el SPGW bastaría con editar su archivo de configuración. El archivo referido es el archivo *spgw.conf*, alojado en el directorio */usr/local/etc/oai*. Este archivo completo se encuentra disponible en el anexo.

Los parámetros que debemos modificar dentro de este, quedando los demás con sus valores por defecto son los siguientes archivo son los siguientes:

Nota: Para su configuración podemos diferenciar el SPGW en los dos componentes que lo forman, por un lado el S-GW y por otro lado el P-GW.

■ S-GW

Incluye los parámetros asociados a las interfaces de red de S-GW:

- **SGW_INTERFACE_NAME_FOR_S11**

Nombre del interfaz S11 mediante el cual nos conectamos al MME.

Le daremos el valor “lo”.

- **SGW_IPV4_ADDRESS_FOR_S11**

Dirección IP del interfaz S11 mediante el cual nos conectamos al MME.

Le daremos el valor “127.0.11.2/8”.

- **SGW_INTERFACE_NAME_FOR_S1U_S12_S4_UP**

Nombre del interfaz mediante el que nos conectaremos al eNB.

Le daremos el valor “eth1”.

- **SGW_IPV4_ADDRESS_FOR_S1U_S12_S4_UP**

Dirección IP del interfaz mediante el que nos conectaremos al eNB.

Le daremos el valor “192.168.1.1/24”.

- **SGW_IPV4_PORT_FOR_S1U_S12_S4_UP**

Puerto de conexión al eNB.

Le daremos el valor “2152”.

■ P-GW

Incluye los parámetros asociados a las interfaces de red de P-GW. En nuestro caso sólo es necesario modificar el siguiente:

● PGW_INTERFACE_NAME_FOR_SGI

Interfaz de red mediante la cual tendremos la salida hacia internet.

Le daremos el valor “eth0”.

5.4. Compilación y puesta en marcha de todo el escenario

En este apartado compilaremos los diferentes componentes que hemos configurado anteriormente y los lanzaremos para comprobar de este modo que han sido correctamente configurados y se conectan correctamente entre sí.

En primer lugar, debemos instalar los certificados necesarios para la compilación puesta en marcha de los equipos componentes de EPC. Lo haremos mediante los comandos:

```
$ cd /openair-cn/SCRIPTS
$ ./check_hss_s6a_certificate /usr/local/etc/oai/freeDiameter/
hss.5GLaboratory
$ ./check_mme_s6a_certificate /usr/local/etc/oai/freeDiameter/
epc.5GLaboratory
```

Una vez instalado los certificados estamos en disposición de compilar todos los paquetes de la red troncal EPC, lo haremos mediante los comandos:

```
$ cd /openair-cn/SCRIPTS
$ ./build_hss -c
$ ./build_mme -c
$ ./build_spgw -c
```

Para lanzar los 3 equipos, debemos hacerlo cada uno en una terminal diferente, y es muy importante que lo hagamos en el orden que se indica, ya

que, si no, podrían aparecer errores.

El primer equipo que debemos lanzar siempre es el HSS, y lo haremos mediante el siguiente comando (suponemos que nos encontramos en el directorio `/openair-cn/SCRIPTS`):

```
$ ./run_hss
```

En segundo lugar lanzamos el MME, estando en el directorio `/openair-cn/SCRIPTS`.

```
$ ./run_mme
```

Y por último, lanzamos el equipo SPGW, estando de nuevo en el directorio `/openair-cn/SCRIPTS`.

```
$ ./run_spgw
```

Una vez tenemos todos los componentes en marcha, podemos comprobar que se ha iniciado correctamente ya que tenemos una nueva interfaz de red en nuestra máquina EPC, como podemos ver en la figura 5.9. Tendremos la dirección IP que especificamos en el archivo de configuración de HSS.

[illegible]

Figura 5.9: Interfaz generada en EPC para el uso de OAI.

Una vez tenemos en marcha nuestra red troncal EPC pasamos a la compilación y puesta en marcha de nuestra red de distribución simulada en la máquina virtual OAISIM.

En primer lugar compilamos OAISIM. Suponemos que nos encontramos en el directorio `/openairinterface5G/cmake-targets`.

```
$ ./build_oai -c -oaisim -UE
```

Hemos seleccionado para la compilación las opciones:

-c, para borrar los archivos temporales y que se haga la compilación desde cero.

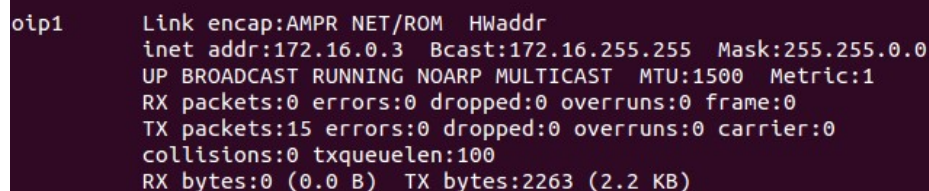
-oaisim, para compilar las partes asociadas a OAISIM.

-UE, para compilar las partes asociadas al UE.

Una vez está compilado estamos en disposición de lanzar la simulación del eNB y del UE, lo haremos en el directorio `/openairinterface5G/cmake-targets/tools` mediante el comando:

```
$ sudo -E ./run_enb_ue_virt_s1
```

Una vez en marcha, podemos comprobar que se ha lanzado correctamente viendo las interfaces de red de las que dispone la máquina, y podemos ver que aparecerá una nueva como la que vemos en la figura 5.10.



```
oip1      Link encap:AMPR NET/ROM  HWaddr  
          inet addr:172.16.0.3  Bcast:172.16.255.255  Mask:255.255.0.0  
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:100  
          RX bytes:0 (0.0 B)  TX bytes:2263 (2.2 KB)
```

Figura 5.10: Interfaz generada en OAISIM para el uso de OAI.

Una vez tenemos esta nueva interfaz de conexión el EPC, podemos comprobar que están conectadas entre sí realizando un *ping* a través de esa interfaz a la máquina EPC. Véase la figura 5.11.

```
$ ping -I oip1 172.16.0.1
```

Mediante el uso de la anterior opción de *ping*, haremos el *ping* que llegará a la máquina EPC forzándolo a que lo haga por la interfaz de funcionamiento de nuestra red OAI.

```
manuel@ubuntu:~$ ping -I oip1 172.16.0.1
PING 172.16.0.1 (172.16.0.1) from 172.16.0.2 oip1: 56(84) bytes of data.
64 bytes from 172.16.0.1: icmp_seq=1 ttl=64 time=745 ms
64 bytes from 172.16.0.1: icmp_seq=2 ttl=64 time=721 ms
64 bytes from 172.16.0.1: icmp_seq=3 ttl=64 time=592 ms
64 bytes from 172.16.0.1: icmp_seq=4 ttl=64 time=1286 ms
64 bytes from 172.16.0.1: icmp_seq=5 ttl=64 time=279 ms
64 bytes from 172.16.0.1: icmp_seq=6 ttl=64 time=653 ms
^C
--- 172.16.0.1 ping statistics ---
7 packets transmitted, 6 received, 14% packet loss, time 6022ms
rtt min/avg/max/mdev = 279.274/713.141/1286.899/299.093 ms, pipe 2
```

Figura 5.11: Prueba de conexión desde OAISIM hacia EPC mediante la interfaz de uso de OAI.

Capítulo 6

Evaluación del escenario

En este capítulo se llevará a cabo el análisis de nuestra red LTE creada mediante el software *OpenAirInterface*.

Nuestro análisis consistirá en una prueba de rendimiento de nuestra red, concretamente nos centraremos en el estudio del ancho de banda soportado por nuestra red cuando se le inyecta tráfico UDP para nuestro eNB configurado con 5, 10 y 20 MHz, para ello debemos modificar el parámetro “N_RB_DL” con los valores 25, 50 y 100 respectivamente. Por último, se estudiarán trazas *wireshark* para comprobar como se realiza la señalización en nuestra red.

6.1. Prueba de rendimiento de la red

6.1.1. Iperf

Para la realización del análisis de rendimiento usaremos la herramienta *iperf* mediante la cual inyectaremos tráfico desde un cliente de la red (en nuestro caso el UE alojado en la máquina virtual OAISIM) hacia un servidor, que en nuestro caso será nuestro núcleo de red, la máquina virtual EPC.

En primer lugar, debemos de instalar la herramienta en ambas VM, para ello lo haremos mediante el uso del siguiente comando:

```
$ sudo apt-get install iperf
```

Una vez la tenemos instalada en ambas máquinas, procedemos a realizar

inyecciones de tráfico UDP en nuestra red.

El primer paso es lanzar un servidor de *iperf* en nuestra VM EPC. Esto lo haremos mediante el uso del siguiente comando.

```
$ iperf -u -s
```

donde será lanzado con las siguientes opciones:

- u, para especificar que el tráfico sea de tipo UDP en lugar de TCP.

- s, para que la herramienta se ponga en marcha en modo servidor.

De este modo, nuestro servidor se quedará escuchando en el canal inyecciones de tráfico UDP.

El segundo paso es inyectar tráfico desde nuestro cliente, para ello haremos uso del comando:

```
$ iperf -u -c 172.16.0.1 -b "ancho de banda inyectado" -B 172.16.0.2
```

En este caso, las opciones elegidas de la herramienta son las siguientes:

- u, para especificar que el tráfico sea de tipo UDP en lugar de TCP.

- c, esta opción tiene la sintaxis -c [host] para que la herramienta se ponga en marcha actuando como cliente, debemos añadirle la dirección IP del equipo al que se conectará, es decir, la dirección IP del servidor. En nuestro caso, será la dirección ip de la interfaz usada por EPC para OAI, 172.16.0.1.

- b, se usa para tráfico UDP, para introducir el ancho de banda que deseamos enviar en bits/seg. En este campo haremos un barrido de valores para obtener en siguientes secciones el rendimiento de nuestra red para diferentes niveles de tráfico ofrecido.

- B, esta opción es del tipo -B [host] y se usa para definir el enlace a [host], una interfaz o una dirección de multidifusión. En nuestro caso añadimos la dirección IP del interfaz de conexión de OAI del cliente, 172.16.0.2.

6.1.2. Resultados obtenidos

Mediante el uso de la herramienta descrita en la sección anterior hemos realizado pruebas variando la opción -b del cliente. De este modo, se ha realizado un barrido de valores comprendidos entre 0.5 Mbits/seg y 700 Mbits/s para cada una de las 3 configuraciones de ancho de banda para eNB (5,10 y 20 MHz). La herramienta nos dará unos valores asociados a la introducción de este tráfico UDP en nuestra red LTE. Estos son *Transfer* (indica la cantidad de datos transferidos) y *Bandwidth* (BW) (indica el ancho de banda real medido en el canal). Estos datos los podemos observar, asociados a cada configuración respectivamente en las tablas 6.1, 6.2 y 6.3.

Ancho de banda inyectado en nuestra red LTE mediante iperf (Mbits/s)	Transfer (MBytes)	Bandwidth (Mbits/s)
0.5	0.613	0.5
1	1.19	1
2	2.39	2
4	4.77	4
6	7.10	5.95
8	9.52	7.99
12	7.10	12
15	17.9	15
20	23.8	20
25	28.4	23.8
30	33.8	28.4
35	39.5	33.1
40	38.7	32.4
50	35.5	29.8
100	36.2	30.3
200	38.2	31.9
400	38.1	31.9
700	39.1	32.8

Tabla 6.1: Datos obtenidos en la prueba de rendimiento de nuestra red LTE para eNB con BW 5MHz

Ancho de banda inyectado en nuestra red LTE mediante iperf (Mbits/s)	Transfer (MBytes)	Bandwich (Mbits/s)
0.5	0.613	0.5
1	1.19	1
2	2.39	2
4	4.77	4
6	7.15	6
8	9.53	7.99
12	14.3	12
15	17.2	14.4
20	22.9	19.2
25	28.8	24.1
30	34.5	28.9
35	36.1	30.2
40	36.7	30.8
50	38.7	32.5
100	40.9	34.3
200	42.5	35.6
400	45.6	38.2
700	49.5	41.4

Tabla 6.2: Datos obtenidos en la prueba de rendimiento de nuestra red LTE para eNB con BW 10MHz

6.1.3. Análisis de resultados

Como podemos ver según los datos obtenidos, a medida que aumentamos el tráfico inyectado en nuestra red LTE, el tráfico cursado por el sistema aumenta de igual manera hasta un determinado punto. A partir de un determinado nivel de tráfico ofrecido, el tráfico cursado en nuestra red se mantiene en un valor aproximadamente constante. Por lo tanto, podemos concluir diciendo que este valor es la capacidad máxima soportada por nuestra red LTE en cada uno de las diferentes configuraciones.

En la figura 6.1 podemos ver de manera más visual en la gráfica los resultados que se acaban de relatar. Para que se aprecie mejor el punto límite donde se alcanza el ancho de banda máximo, se han representado los valores comprendidos en el intervalo (0.5 , 700) Mbits/s del tráfico inyectado mediante *iperf*.

Ancho de banda inyectado en nuestra red LTE mediante iperf (Mbits/s)	Transfer (MBytes)	Bandwich (Mbits/s)
0.5	0.613	0.5
1	1.19	1
2	2.39	2
4	4.77	4
6	7.11	5.97
8	9.52	7.99
12	14.1	11.8
15	17.8	14.9
20	23.3	19.5
25	27.8	23.3
30	32.3	27.1
35	36.2	30.4
40	39.7	33.3
50	38	31.8
100	54.5	45.6
200	66.9	56.1
400	68	57
700	73.7	61.8

Tabla 6.3: Datos obtenidos en la prueba de rendimiento de nuestra red LTE para eNB con BW 20MHz

Teóricamente, el resultado que debemos obtener de rendimiento debe ser proporcional al ancho de banda configurado en el eNB (5, 10, 20 MHz). Sin embargo, el rendimiento no óptimo de las VMs provoca que no se consiga realizar la transmisión a toda la velocidad posible por las restricciones del PC. Es por eso por lo que no se consiguen unos resultados todo lo correctos que se podrían.

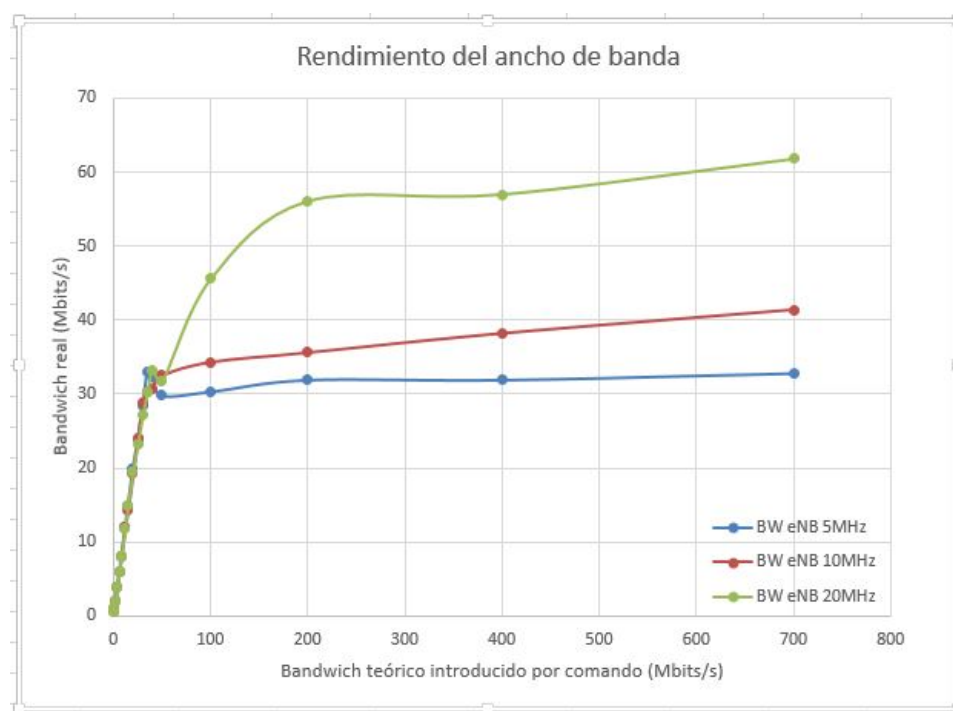


Figura 6.1: Curva de rendimiento del ancho de banda en nuestra red LTE para nuestros 3 tipos de configuración del ancho del BW de eNB

6.2. Señalización

En esta sección, mediante el uso de la herramienta *wireshark* comprobaremos cómo se realiza la señalización de los paquetes cuando se conecta nuestras VM OAISIM y EPC.

Para poder realizar esta prueba, en primer lugar debemos lanzar el EPC (con todas sus partes en ejecución), una vez está corriendo, lanzamos *wireshark* en cualquiera de las dos máquinas y lanzamos OAISIM como se indica en el capítulo 6.

Una vez hemos realizado la captura de trazas con *wireshark* debemos filtrar por protocolo para centrarnos en los mensajes de señalización, concretamente este protocolo es S1-AP. Podemos observar en la figura 6.2, tras introducir el filtro, todos los mensajes de señalización que se han producido.

No.	Time	Source	Destination	Protocol	Length	Info
152	8.437423000	192.168.1.1	192.168.1.2	SLAP	124	id-S1Setup, S1SetupRequest
161	88.587763000	192.168.1.1	192.168.1.2	SLAP	92	id-S1Setup, S1SetupResponse
165	95.723698000	192.168.1.2	192.168.1.1	SLAP/NAS-	160	id-InitialUEMessage, Attach request, PDN connectivity request
178	95.853378000	192.168.1.2	192.168.1.2	SLAP/NAS-	148	SACK id-downlinkNASTransport, Authentication request
188	97.828479000	192.168.1.2	192.168.1.1	SLAP/NAS-	128	id-uplinkNASTransport, Authentication response
181	97.839553000	192.168.1.1	192.168.1.2	SLAP/NAS-	124	SACK id-downlinkNASTransport, Security mode command
183	97.694472000	192.168.1.2	192.168.1.1	SLAP/NAS-	124	id-uplinkNASTransport, Security mode complete
201	97.769488000	192.168.1.1	192.168.1.2	SLAP/NAS-	288	SACK id-InitialContextSetup, InitialContextSetupRequest , Attach accept, Activate default EPS bearer context request
203	98.952017000	192.168.1.2	192.168.1.1	SLAP	116	id-UECapabilityInfoIndicationUECapabilityInformation
205	99.707376000	192.168.1.2	192.168.1.1	SLAP	168	id-InitialContextSetup, InitialContextSetupResponse
209	99.987181000	192.168.1.2	192.168.1.1	SLAP/NAS-	128	id-uplinkNASTransport, Attach complete, Activate default EPS bearer context accept

Figura 6.2: Mensajes de señalización entre nuestras VM.

Pasamos ahora a ver qué señaliza cada mensaje así como la información relevante que puedan tener en su interior.

6.2.1. Señalización de la conexión eNB

En primer lugar, podemos observar el primer par de mensajes que aparecen en la figura 6.2. Consisten en la solicitud de conexión desde nuestra VM OASIM, en concreto desde el eNB, hacia nuestra VM EPC, en concreto con el MME, y su respuesta de conexión inversa.

En el primero de estos dos mensajes, llamado “*S1SetupRequest*” se envía la información la información global del ENB que se va a generar, como su MCC, su MNC, su nombre de identificación, etcétera. Podemos verlo en la figura 6.3

▼S1SetupRequest
▼protocolIEs: 4 items
▼Item 0: id-Global-ENB-ID
▼ProtocolIE-Field
id: id-Global-ENB-ID (59)
criticality: reject (0)
▼value
▼Global-ENB-ID
plMNidentity: 02f839
Mobile Country Code (MCC): France (208)
Mobile Network Code (MNC): Unknown (93)
▶eNB-ID: macroENB-ID (0)
▼Item 1: id-eNBname
▼ProtocolIE-Field
id: id-eNBname (60)
criticality: ignore (1)
▼value
0... Extension Present Bit: False
ENBname: eNB_Eurecom_LTEBox
▶Item 2: id-SupportedTAs
▶Item 3: id-DefaultPagingDRX

Figura 6.3: Información contenida en el mensaje S1SetupRequest

En el segundo mensaje, llamado “*S1SetupResponse*” se envía la respuesta al mensaje anterior. Podemos ver su contenido en la figura 6.4.

```

▼ S1SetupResponse
  ▼ protocolIEs: 2 items
    ▼ Item 0: id-ServedGUMMEIs
      ▼ ProtocolIE-Field
        id: id-ServedGUMMEIs (105)
        criticality: reject (0)
        ▼ value
          ▼ ServedGUMMEIs: 1 item
            ▼ Item 0
              ▼ ServedGUMMEIsItem
                ▼ servedPLMNs: 1 item
                  ▼ Item 0
                    PLMNidentity: 02f839
                    Mobile Country Code (MCC): France (208)
                    Mobile Network Code (MNC): Unknown (93)
                ▼ servedGroupIDs: 1 item
                  ▼ Item 0
                    MME-Group-ID: 0004
                ▼ servedMMECs: 1 item
                  ▼ Item 0
                    MME-Code: 01
            ► Item 1: id-RelativeMMECapacity

```

Figura 6.4: Información contenida en el mensaje S1SetupResponse

6.2.2. Señalización de la conexión UE

Mediante el mensaje “*InitialUEMessage*” enviado desde OAISIM al EPC para mandarle la información al MME, y éste le asigna un identificador para incluir el UE en sus tablas de enrutamiento.

El mensaje incluye como datos el identificador del UE, los protocolos soportados, el MCC, el MNC, y posiblemente el más importante que es el IMSI asociado al UE. Se puede ver su contenido en la figura 6.5.

6.2.3. Señalización de autenticación y seguridad

Los mensajes de señalización y autenticación son los que aparecen en la figura 6.6.

El primero de los mensajes llamado “*Attach request*”, es el que señala el comienzo del duelo para realizar la autenticación. Y su mensaje pareado, el cual pone fin al duelo es el último, el llamado “*Attach complete*” mediante el cual se da por completada la autenticación.

El mensaje “*Authentication Request*” es enviado desde el MME al eNB para que lo haga llegar al UE solicita el proceso de autenticación y se envían en este los parámetros RAND y AUTN. Podemos observarlo en la figura 6.7. Por otro lado el mensaje “*Authentication Response*” que es la respuesta al mensaje anterior, donde nuestro UE envía como respuesta al mensaje de autenticación con el parámetro de autenticación RES. Podemos observarlo

id: id-ENB-UE-SIAP-ID (8)	
criticality: reject (0)	
▼value	
ENB-UE-SIAP-ID: 4367919	
▼Item 1: id-NAS-PDU	
▼ProtocolIE-Field	
id: id-NAS-PDU (26)	
criticality: reject (0)	
▼value	
NAS-PDU: 0741710829803910000011110280200200201d011271a80...	
▼Non-Access-Stratum (NAS)PDU	
0000 = Security header type: Plain NAS message, not security protected (0)	
.... 0111 = Protocol discriminator: EPS mobility management messages (0x07)	
NAS EPS Mobility Management Message Type: Attach request (0x41)	
0... = Type of security context flag (TSC): Native security context (for KSIasme)	
.111 = NAS key set identifier: No key is available (7)	
.... 0... = Spare bit(s): 0x00	
.... .001 = EPS attach type: EPS attach (1)	
▼EPS mobile identity	
Length: 8	
.... 1... = odd/even indic: 1	
.... .001 = Type of identity: IMSI (1)	
IMSI: 208930100001111	
►UE network capability	
►ESM message container	
▼Item 2: id-TAI	
▼ProtocolIE-Field	
id: id-TAI (67)	
criticality: reject (0)	

Figura 6.5: Información contenida en el mensaje InitialUEMessage

165 95.72236000e192.168.1.2	192.168.1.1	SIAP/NAS-	160 id-initialUEMessage, Attach request, PDN connectivity request
178 95.85337800e192.168.1.1	192.168.1.2	SIAP/NAS-	140 SACK id-downlinkNASTransport, Authentication request
180 97.02047900e192.168.1.2	192.168.1.1	SIAP/NAS-	128 id-uplinkNASTransport, Authentication response
181 97.03955200e192.168.1.1	192.168.1.2	SIAP/NAS-	124 SACK id-downlinkNASTransport, Security mode command
183 97.09447200e192.168.1.2	192.168.1.1	SIAP/NAS-	124 id-uplinkNASTransport, Security mode complete
201 97.76948800e192.168.1.1	192.168.1.2	SIAP/NAS-	280 SACK id-InitialContextSetup, InitialContextSetupRequest, Attach accept, Activate default EPS bearer context request
209 99.90718100e192.168.1.2	192.168.1.1	SIAP/NAS-	128 id-uplinkNASTransport, Attach complete, Activate default EPS bearer context accept

Figura 6.6: Mensajes de señalización de autenticación y seguridad

en la figura 6.8.

ENB-UE-SIAP-ID: 4367919	
▼Item 2: id-NAS-PDU	
▼ProtocolIE-Field	
id: id-NAS-PDU (26)	
criticality: reject (0)	
▼value	
NAS-PDU: 075200e23daf49006179616f637661fedf206010ca543e79...	
▼Non-Access-Stratum (NAS)PDU	
0000 = Security header type: Plain NAS message, not security protected (0)	
.... 0111 = Protocol discriminator: EPS mobility management messages (0x07)	
NAS EPS Mobility Management Message Type: Authentication request (0x52)	
0000 = Spare half octet: 0	
.... 0... = Type of security context flag (TSC): Native security context (for KSIasme)	
.... .000 = NAS key set identifier: (0) ASME	
▼Authentication Parameter RAND - EPS challenge	
RAND value: e23daf49006179616f637661fedf2060	
▼Authentication Parameter AUTN (UMTS and EPS authentication challenge) - EPS challenge	
Length: 16	
▼AUTN value: ca543e79c2658000ff512e09481833a8	
SQN xor AK: ca543e79c265	
AMF: 8000	
MAC: ff512e09481833a8	

Figura 6.7: Información contenida en el mensaje Authentication Request



Figura 6.8: Información contenida en el mensaje Authentication Response

Tras la recepción del mensaje “*Authentication Response*” se da por concluida la autenticación, el MME vuelve a enviar un mensaje al UE donde se indican los algoritmos de seguridad que se van a emplear para la comunicación en el mensaje “*Security Mode Command*”. El mensaje de respuesta en dirección inversa al anterior es “*Security Mode Complete*”, donde se certifica la activación de los parámetros de seguridad.

El último mensaje que nos faltaría citar de los que aparecen en la figura 6.6 es el “*InitialContextSetup*” mediante el cual, una vez establecidos los procesos de autenticación, se le envía desde el MME al UE las especificaciones para que éste se pueda conectar a la red. Contiene en su interior información como el nombre del punto de acceso, la dirección IP asociada al UE, la dirección IP del S-GW, el *Globally Unique Temporary Identity*. (GUTI), el identificador de QoS que usará el UE, el *bit rate*, la clave de seguridad, etcétera.

Este mensaje es respondido con el mensaje “*Initial Context Setup Response*” mediante el cual se indicará la dirección IP del eNB donde está localizado nuestro UE.


```

NAS EPS Mobility Management Message Type: Attach accept (0x42)
0000 .... = Spare half octet: 0
.... 0... = Spare bit(s): 0x00
.... 010 = Attach result: Combined EPS/IMSI attach (2)
▶ GPRS Timer - T3412 value
▶ Tracking area identity list - TAI list
▼ ESM message container
  Length: 56
▼ ESM message container contents: 5201c1010909036f616904697076340501ac1000025e04fe...
  0101 .... = EPS bearer identity: EPS bearer identity value 5 (5)
  .... 0010 = Protocol discriminator: EPS session management messages (0x02)
  Procedure transaction identity: 1
  NAS EPS session management messages: Activate default EPS bearer context request (0xc1)
  ▶ EPS quality of service
  ▼ Access Point Name
    Length: 9
    APN: oai.ipv4
  ▼ PDN address
    Length: 5
    0000 0... = Spare bit(s): 0x00
    PDN type: IPv4 (1)
    PDN IPv4: 172.16.0.2 (172.16.0.2)
    ▶ APN aggregate maximum bit rate
    ▶ Protocol Configuration Options
  ▶ EPS mobile identity - GUTI
  ▶ GPRS Timer - T3402 value
▶ Item 4: id-UESecurityCapabilities
▶ Item 5: id-SecurityKey

```

Figura 6.9: Información contenida en el mensaje InitialContextSetup

Capítulo 7

Conclusiones

Este capítulo pone fin a la memoria del presente proyecto. Durante este capítulo se exponen las principales conclusiones a las que se ha llegado mediante al realización del proyecto al completo. También se exponen las líneas futuras mediante las cuales podrían mejorarse el proyecto. Y, por último, se incluye una valoración personal sobre el trabajo realizado.

7.1. Conclusión

En este proyecto el principal objetivo buscado era la familiarización y estudio del funcionamiento real (aunque finalmente se ha realizado mediante emulación) de una tecnología de redes móviles en pleno rendimiento en la actualidad como es LTE.

Para conseguir esto, nos planteamos como objetivo el montaje de una red LTE mediante un SDR usando un *software* como *Open Air Interface* donde, por un lado generaríamos la red troncal (EPC) y, por otro lado, implementaríamos la red de acceso radio E-UTRAN. Como EPC se emplearía una máquina virtual que incluye sus tres componentes principales que son el MME, el HSS y SPGW (que engloba dentro de sí el S-GW y el P-GW). Para la red E-UTRAN se usaría otra máquina virtual que implementaría un eNB y un UE en su interior. Por último, se pondría a prueba nuestra red una vez realizado el despliegue al completo de ésta, realizando un análisis de rendimiento, midiendo su latencia y comprobando mediante trazas de *wireshark* como realiza la señalización.

Tras la finalización de este proyecto, podemos dar una serie de ventajas e inconvenientes a la herramienta que hemos usado:

- La implementación de una red LTE mediante el *software Open Air Interface* es económica. Por lo tanto, es muy recomendable para su uso de carácter didáctico aunque, sin embargo, quizás no disponga de un rendimiento suficientemente alto como para realizar una implementación real.
- La versión utilizada de *Open Air Interface* no es la más actualizada, debido a que se ha utilizado una versión donde la emulación funcionaba correctamente. Si hubiésemos dispuesto de los equipos SDR sí se habría utilizado la última versión, ya que se habría podido implementar una red real.
- Para este escenario en concreto, sí que había mucha información al respecto de su configuración y puesta en marcha. Sin embargo, para otro tipo de escenarios, aunque la OSA muestra su funcionamiento mediante vídeos o imágenes en sus medios de comunicación, no está disponible su código aún. Esto ocurre, por ejemplo, para versiones que implementan *Narrowband-IoT* (NB-IoT) y 5G NR.

7.2. Líneas futuras

Algunas de las líneas futuras que podría adaptar el siguiente proyecto son las siguientes:

- Podríamos sobredimensionar nuestra red, por ejemplo, mediante la implementación de más UEs para ver cómo se comportaría nuestra red ante un mayor número de abonados, como sería una red real comercial. También podríamos añadir algún eNB más para acoplarlos a nuestra red, crear diferentes celdas y ver cómo realizarían nuestros abonados el intercambio de celda según su ubicación en el espacio, etcétera.
- Implementación del escenario actual, separando físicamente en diferentes PCs las distintas partes de la red LTE, usando un USRP para la transmisión de señal y usar como dispositivo final de usuario un UE comercial real. Así, se podrá analizar en este escenario la señal enviada, la señal recibida, las condiciones radio, etcétera. Esta implementación era la que al comienzo del proyecto se pretendía realizar pero finalmente se vio modificada.
- Implementación de escenario que se tenía previsto realizar al comienzo de la planificación de este TFG, es decir, la implementación de un escenario desplegado con equipos físicos que implemente *Cloud RAN*.

- Podría implementarse también otro tipo de escenario cuyo objetivo podría ser familiarizarse con alguna diferente técnica del estándar LTE de 3GPP, o incluso con otro estándar de red móvil, como podría ser el estándar 5G.

7.3. Valoración personal

La elección de este proyecto se basa en mi gran interés desde hace años por el campo de las redes de telecomunicaciones y en concreto de las redes móviles. Es una de los campos que más me gusta de las telecomunicaciones y posiblemente fue una de las mayores motivaciones que me llevaron a elegir cursar este grado.

La realización de este proyecto ha supuesto la recta final de mis estudios de grado, y ha supuesto para mí un gran reto tanto personal como profesional por los diferentes inconvenientes y dificultades que se han ido planteando a lo largo de su desarrollo, pero si algo he aprendido durante estos últimos años de formación es a anteponerme a situaciones adversas y poco a poco ir consiguiendo los objetivos propuestos.

Desde aquella primera charla que mantuve con mi tutor a finales del pasado curso para comenzar a realizar mi TFG durante el presente, he sentido una gran motivación por su desarrollo y por el fin de lograr los objetivos que planteamos aquel día. Ha sido una pena tener que, a mitad del camino, modificar los objetivos planteados en un principio, aunque el objetivo principal que siempre ha sido profundizar mis conocimientos sobre las redes móviles y en este caso en concreto, en el estándar LTE, lo he conseguido y por ello estoy muy satisfecho tanto a nivel personal como profesional con mi trabajo realizado en este proyecto.

Por último, debemos destacar, que si algo hemos aprendido durante este desgraciada pandemia vivido a nivel mundial, que ha sucedido sucedido durante el desarrollo de este proyecto, es que lo más importante siempre es y será la salud, por encima de cualquier aspecto profesional y que a pesar de no haber podido realizar el proyecto tal y como deseaba me siento muy afortunado de no haber sufrido este virus ni yo ni ninguno de mis seres queridos. Desde aquí me gustaría enviar mi agradecimiento a todas las personas que han estado luchado y lo hacen cada en primera línea por la salud y bienestar de toda la sociedad, y expresar mis condolencias a todas las personas que por culpa de este virus han perdido algún ser querido.

Bibliografía

- [1] *3GPP*. <https://www.3gpp.org>. Accedido en julio de 2020.
- [2] Dahlman, Erik; Parkvall, Stefan; Sköld Johan: *4G LTE/LTE-Advanced for Mobile Broadband*. Amsterdam ; Boston : Elsevier/Academic Press, 2011.
- [3] IPv6 Go: *Tutorial LTE. Arquitectura de una red LTE*. http://www.ipv6go.net/lte/arquitectura_red_lte.php. Accedido en julio de 2020.
- [4] *Khomp*. <https://www.khomp.com/>. Accedido en julio de 2020.
- [5] Alvarez-Campana, Manuel: *Curso LTE. Arquitectura funcional y protocolos*. <http://catedraisdefe.etsit.upm.es/wp-content/uploads/2015/04/Manuel-Alvarez-Campana-T3.pdf>. Accedido en julio de 2020.
- [6] Revolution WI-FI: *PHY Basics: How OFDM Subcarriers Work*. <http://www.revolutionwifi.net/revolutionwifi/2015/3/how-ofdm-subcarriers-work>. Accedido en julio de 2020.
- [7] Trivedi, Vinay Kumar: *Carrier Interferometry Coded Single Carrier FDMA (CI/SC-FDMA) for Next Generation Underwater Acoustic Communication*. <https://link.springer.com/article/10.1007/s11277-017-4119-1#citeas>. Accedido en julio de 2020.
- [8] *Open Air Interface*. <https://www.openairinterface.org/>. Accedido en julio de 2020.
- [9] *EURECOM*. <http://www.eurecom.fr/>. Accedido en julio de 2020.
- [10] García, Rocío: *Todas las OMV en España que puedes contratar, ventajas y cobertura*. <https://www.adslzone.net/reportajes/operadores/listado-omv-espana>. Accedido en julio de 2020.
- [11] *Concesiones de frecuencias de telefonía móvil*. Nota de prensa, MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO, Julio

2013. <https://www.mincotur.gob.es/es-es/gabineteprensa/notasprensa/2013/documents/npextensionconcesionesm%C3%B3viles030713.pdf>.
- [12] Empresa, Ministerio de Economía y: *Cobertura de Banda Ancha en España en el año 2018*. <https://avancedigital.gob.es/banda-ancha/cobertura/Documents/Cobertura-BA-2018.pdf>. Accedido en julio de 2020.
- [13] 5G. Ministerio de asuntos económicos y transformación digital. <https://avancedigital.gob.es/5G/Paginas/Index.aspx>. Accedido en julio de 2020.
- [14] UGR: *Plan de contingencia Covid-19*. <https://covid19.ugr.es/informacion/plan-contingencia>. Accedido en julio de 2020.
- [15] Wolfe, Mike: *CommScope Definitions: What is C-RAN?* <https://www.commscope.com/Blog/CommScope-Definitions-What-is-C-RAN>. Accedido en julio de 2020.
- [16] *srsLTE*. <https://www.srslte.com>. Accedido en julio de 2020.
- [17] Gomez-Miguel, Ismael; Garcia-Saavedra, Andres; Sutton Paul; Serrano Pablo; Cano Cristina; Leith Douglas: *srsLTE: An Open-Source Platform for LTE Evolution and Experimentation*. Octubre 2016.
- [18] *Amarisoft*. <https://www.amarisoft.com>. Accedido en julio de 2020.
- [19] *ASUS*. <https://www.asus.com>. Accedido en julio de 2020.
- [20] *DELL*. <https://www.dell.com>. Accedido en julio de 2020.
- [21] *Fujitsu*. <https://www.fujitsu.com>. Accedido en julio de 2020.
- [22] *SAMSUNG*. <https://www.samsung.com>. Accedido en julio de 2020.
- [23] *National Instrument*. <https://www.ni.com>. Accedido en julio de 2020.
- [24] *Ettus*. <https://www.ettus.com>. Accedido en julio de 2020.
- [25] *Sysmocom*. <http://shop.sysmocom.de>. Accedido en julio de 2020.
- [26] *Ubuntu*. <https://ubuntu.com>. Accedido en julio de 2020.
- [27] *Windows*. <https://www.microsoft.com/es-es/windows>. Accedido en julio de 2020.
- [28] *Grantt Project*. <https://www.ganttproject.biz>. Accedido en julio de 2020.

-
- [29] *Github*. <https://github.com>. Accedido en julio de 2020.
 - [30] *Overleaf*. <https://www.overleaf.com>. Accedido en julio de 2020.
 - [31] *Lucidchart*. <https://app.lucidchart.com>. Accedido en julio de 2020.
 - [32] *Wireshark*. <https://www.wireshark.org>. Accedido en julio de 2020.
 - [33] *Oracle VM VirtualBox*. <https://www.virtualbox.org>. Accedido en julio de 2020.
 - [34] *Iperf*. <https://iperf.fr>. Accedido en julio de 2020.
 - [35] *Microsoft Office*. <https://www.microsoft.com/es-es/microsoft-365>. Accedido en julio de 2020.
 - [36] Cox, Christopher: *An introduction to LTE LTE, LTE-advanced, SAE, and 4G mobile communications*. Hoboken, N.J.: John Wiley & Sons, 2nd edición, 2012.
 - [37] Ahmed, Bannour; Abdul Matin, Mohammad: *Coding for MIMO-OFDM in Future Wireless Systems*. Cham : Springer International Publishing : Imprint: Springer, 1st ed. edición, 2015.

Apéndice A

Archivos de configuración de EPC

A.1. hss.conf

```
1 #####
2 # Licensed to the OpenAirInterface (OAI) Software Alliance under one
3 # contributor license agreements. See the NOTICE file distributed
4 # with
5 # this work for additional information regarding copyright ownership.
6 # The OpenAirInterface Software Alliance licenses this file to You
7 # under
8 # the Apache License, Version 2.0 (the "License"); you may not use
9 # this file
10 # except in compliance with the License.
11 # You may obtain a copy of the License at
12 #
13 # http://www.apache.org/licenses/LICENSE-2.0
14 #
15 # Unless required by applicable law or agreed to in writing, software
16 # distributed under the License is distributed on an "AS IS" BASIS,
17 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
18 # implied.
19 # See the License for the specific language governing permissions and
20 # limitations under the License.
21 #-----
22 # For more information about the OpenAirInterface (OAI) Software
23 # Alliance:
24 # contact@openairinterface.org
25 #####
26 HSS :
27 {
28 ## MySQL mandatory options
29 MYSQL_server = "127.0.0.1"; # HSS S6a bind address
30 MYSQL_user = "root"; # Database server login
31 MYSQL_pass = "5GLaboratory"; # Database server password
32 MYSQL_db = "oai_db"; # Your database name
```

```
28
29 ## HSS options
30 OPERATOR_key = "1006020f0a478bf6b699f15c062e42b3"; # OP key matching
    your database
31 #OPERATOR_key = "11111111111111111111111111111111"; # OP key matching
    your database
32
33 RANDOM = "true";                                # True random or
    only pseudo random (for subscriber vector generation)
34
35 ## Freediameter options
36 FD_conf = "/usr/local/etc/oai/freeDiameter/hss_fd.conf";
37 };
```

A.2. hss_fd.conf

```
1 # ----- Local -----
2 # The first parameter in this section is Identity, which will be used
   to
3 # identify this peer in the Diameter network. The Diameter protocol
   mandates
4 # that the Identity used is a valid FQDN for the peer. This parameter
   can be
5 # omitted, in that case the framework will attempt to use system
   default value
6 # (as returned by hostname --fqdn).
7 Identity = "hss.5GLaboratory";
8
9 # In Diameter, all peers also belong to a Realm. If the realm is not
   specified,
10 # the framework uses the part of the Identity after the first dot.
11 Realm = "5GLaboratory";
12
13
14 # This parameter is mandatory, even if it is possible to disable TLS
   for peers
15 # connections. A valid certificate for this Diameter Identity is
   expected.
16 TLS_Cred = "/usr/local/etc/oai/freeDiameter/hss.cert.pem", "/usr/local
   /etc/oai/freeDiameter/hss.key.pem";
17 TLS_CA = "/usr/local/etc/oai/freeDiameter/hss.cacert.pem";
18
19
20 # Disable use of TCP protocol (only listen and connect in SCTP)
21 # Default : TCP enabled
22 No_SCTP;
23
24
25 # This option is ignored if freeDiameter is compiled with DISABLE_SCTP
   option.
26 # Prefer TCP instead of SCTP for establishing new connections.
27 # This setting may be overwritten per peer in peer configuration blocs
   .
28 # Default : SCTP is attempted first.
29 Prefer_TCP;
30
31
32 # Disable use of IPv6 addresses (only IP)
33 # Default : IPv6 enabled
34 No_IPv6;
35
36
37 # Overwrite the number of SCTP streams. This value should be kept low,
38 # especially if you are using TLS over SCTP, because it consumes a lot
   of
39 # resources in that case. See tickets 19 and 27 for some additional
   details on
40 # this.
41 # Limit the number of SCTP streams
42 SCTP_streams = 3;
43
44
45 # By default, freeDiameter acts as a Diameter Relay Agent by
   forwarding all
```

```

46 # messages it cannot handle locally. This parameter disables this
    behavior.
47 NoRelay;
48
49
50 # Use RFC3588 method for TLS protection, where TLS is negotiated after
    CER/CEA exchange is completed
51 # on the unsecure connection. The alternative is RFC6733 mechanism,
    where TLS protects also the
52 # CER/CEA exchange on a dedicated secure port.
53 # This parameter only affects outgoing connections.
54 # The setting can be also defined per-peer (see Peers configuration
    section).
55 # Default: use RFC6733 method with separate port for TLS.
56
57 #TLS_old_method;
58
59
60 # Number of parallel threads that will handle incoming application
    messages.
61 # This parameter may be deprecated later in favor of a dynamic number
    of threads
62 # depending on the load.
63 AppServThreads = 4;
64
65 # Specify the addresses on which to bind the listening server. This
    must be
66 # specified if the framework is unable to auto-detect these addresses,
    or if the
67 # auto-detected values are incorrect. Note that the list of addresses
    is sent
68 # in CER or CEA message, so one should pay attention to this parameter
    if some
69 # addresses should be kept hidden.
70 #ListenOn = "127.0.0.1";
71
72 Port = 3868;
73 SecPort = 5868;
74
75 # ----- Extensions -----
76
77 # Uncomment (and create rtd.conf) to specify routing table for this
    peer.
78 #LoadExtension = "rt_default.fdx" : "rtd.conf";
79
80 # Uncomment (and create acl.conf) to allow incoming connections from
    other peers.
81 LoadExtension = "acl_wl.fdx" : "/usr/local/etc/oai/freeDiameter/acl.
    conf";
82
83 # Uncomment to display periodic state information
84 #LoadExtension = "dbg_monitor.fdx";
85
86 # Uncomment to enable an interactive Python interpreter session.
    # (see doc/dbg_interactive.py.sample for more information)
87 #LoadExtension = "dbg_interactive.fdx";
88
89
90 # Load the RFC4005 dictionary objects
91 #LoadExtension = "dict_nasreq.fdx";
92
93 LoadExtension = "dict_nas_mip6.fdx";
94 LoadExtension = "dict_s6a.fdx";

```

```
95
96 # Load RFC4072 dictionary objects
97 #LoadExtension = "dict_eap.fdx";
98
99 # Load the Diameter EAP server extension (requires diameap.conf)
100 #LoadExtension = "app_diameap.fdx" : "diameap.conf";
101
102 # Load the Accounting Server extension (requires app_acct.conf)
103 #LoadExtension = "app_acct.fdx" : "app_acct.conf";
104
105 # ----- Peers -----
106
107 # The framework will actively attempt to establish and maintain a
   connection
108 # with the peers listed here.
109 # For only accepting incoming connections, see the acl_wl.fx extension
   .
110
111 #ConnectPeer = "epc.5GLaboratory" { ConnectTo = "127.0.0.1"; No_TLS;
   };
```

A.3. mme.conf

```

1 #####
2 # Licensed to the OpenAirInterface (OAI) Software Alliance under one
   or more
3 # contributor license agreements. See the NOTICE file distributed
   with
4 # this work for additional information regarding copyright ownership.
5 # The OpenAirInterface Software Alliance licenses this file to You
   under
6 # the Apache License, Version 2.0 (the "License"); you may not use
   this file
7 # except in compliance with the License.
8 # You may obtain a copy of the License at
9 #
10 #    http://www.apache.org/licenses/LICENSE-2.0
11 #
12 # Unless required by applicable law or agreed to in writing, software
13 # distributed under the License is distributed on an "AS IS" BASIS,
14 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied.
15 # See the License for the specific language governing permissions and
16 # limitations under the License.
17 #-----
18 # For more information about the OpenAirInterface (OAI) Software
   Alliance:
19 #    contact@openairinterface.org
20 #####
21
22 MME :
23 {
24     REALM                                = "5GLaboratory";
                                   # YOUR REALM HERE
25     # Define the limits of the system in terms of served eNB and
   served UE.
26     # When the limits will be reached, overload procedure will take
   place.
27     MAXENB                                = 2;
                                   # power of 2
28     MAXUE                                = 16;
                                   # power of 2
29     RELATIVE_CAPACITY                    = 10;
30
31     EMERGENCY_ATTACH_SUPPORTED            = "no";
32     UNAUTHENTICATED_IMSI_SUPPORTED        = "no";
33
34     # EPS network feature support
35     EPS_NETWORK_FEATURE_SUPPORT_IMS_VOICE_OVER_PS_SESSION_IN_S1    =
   "no";    # DO NOT CHANGE
36     EPS_NETWORK_FEATURE_SUPPORT_EMERGENCY_BEARER_SERVICES_IN_S1_MODE =
   "no";    # DO NOT CHANGE
37     EPS_NETWORK_FEATURE_SUPPORT_LOCATION_SERVICES_VIA_EPC            =
   "no";    # DO NOT CHANGE
38     EPS_NETWORK_FEATURE_SUPPORT_EXTENDED_SERVICE_REQUEST              =
   "no";    # DO NOT CHANGE
39
40     # Display statistics about whole system (expressed in seconds)
41     MME_STATISTIC_TIMER                = 10;
42

```



```

43 IP_CAPABILITY = "IPv4v6";                                     # UNUSED,
    TODO
44
45
46 INTERTASK_INTERFACE :
47 {
48     # max queue size per task
49     ITTI_QUEUE_SIZE          = 2000000;
50 };
51
52 S6A :
53 {
54     S6A_CONF                  = "/usr/local/etc/oai/freeDiameter/
    mme_fd.conf"; # YOUR MME freeDiameter config file path
55     HSS_HOSTNAME              = "hss";
    # THE HSS HOSTNAME
56 };
57
58 # ----- SCTP definitions
59 SCTP :
60 {
61     # Number of streams to use in input/output
62     SCTP_INSTREAMS = 8;
63     SCTP_OUTSTREAMS = 8;
64 };
65
66 # ----- S1AP definitions
67 S1AP :
68 {
69     # outcome drop timer value (seconds)
70     S1AP_OUTCOME_TIMER = 10;
71 };
72
73 # ----- MME served GUMMEIs
74 # MME code DEFAULT size = 8 bits
75 # MME GROUP ID size = 16 bits
76 GUMMEI_LIST = (
77     {MCC="208" ; MNC="93"; MME_GID="4" ; MME_CODE="1"; }
    # YOUR GUMMEI CONFIG HERE
78 );
79
80 # ----- MME served TAIs
81 # TA (mcc.mnc:tracking area code) DEFAULT = 208.34:1
82 # max values = 999.999:65535
83 # maximum of 16 TAIs, comma separated
84 # !!! Actually use only one PLMN
85 TAI_LIST = (
86     {MCC="208" ; MNC="93"; TAC = "1"; }
    # YOUR TAI CONFIG HERE
87 );
88
89
90 NAS :
91 {
92     # 3GPP TS 33.401 section 7.2.4.3 Procedures for NAS algorithm
    selection
93     # decreasing preference goes from left to right
94     ORDERED_SUPPORTED_INTEGRITY_ALGORITHM_LIST = [ "EIA2" , "EIA1"
    , "EIA0" ];
95     ORDERED_SUPPORTED_CIPHERING_ALGORITHM_LIST = [ "EEA0" , "EEA1"
    , "EEA2" ];

```

[illegible]

[illegible]

```

177     MME_IPV4_ADDRESS_FOR_S11_MME           = "127.0.11.1/8";
178         # YOUR NETWORK CONFIG HERE
179     MME_PORT_FOR_S11_MME                   = 2123;
180         # YOUR NETWORK CONFIG HERE
181 };
182
183 LOGGING :
184 {
185     # OUTPUT choice in { "CONSOLE", "SYSLOG", 'path to file', "'
186     # IPv4@': 'TCP port num'"}
187     # 'path to file' must start with '.' or '/'
188     # if TCP stream choice, then you can easily dump the traffic
189     # on the remote or local host: nc -l 'TCP port num' >
190     # received.txt
191     OUTPUT                                = "CONSOLE";
192     #OUTPUT                               = "SYSLOG";
193     #OUTPUT                               = "/tmp/mme.log";
194     #OUTPUT                               = "127.0.0.1:5656";
195
196     # THREAD_SAFE choice in { "yes", "no" } means use of thread
197     # safe intermediate buffer then a single thread pick each
198     # message log one
199     # by one to flush it to the chosen output
200     THREAD_SAFE                           = "no";
201
202     # COLOR choice in { "yes", "no" } means use of ANSI styling
203     # codes or no
204     COLOR                                 = "yes";
205
206     # Log level choice in { "EMERGENCY", "ALERT", "CRITICAL", "
207     # ERROR", "WARNING", "NOTICE", "INFO", "DEBUG", "TRACE"}
208     SCTP_LOG_LEVEL                        = "NOTICE";
209     S11_LOG_LEVEL                         = "TRACE";
210     GTPV2C_LOG_LEVEL                     = "TRACE";
211     UDP_LOG_LEVEL                        = "TRACE";
212     S1AP_LOG_LEVEL                       = "TRACE";
213     NAS_LOG_LEVEL                        = "TRACE";
214     MME_APP_LOG_LEVEL                    = "TRACE";
215     S6A_LOG_LEVEL                        = "TRACE";
216     SECU_LOG_LEVEL                       = "TRACE";
217     UTIL_LOG_LEVEL                       = "TRACE";
218     MSC_LOG_LEVEL                        = "WARNING";
219     ITTI_LOG_LEVEL                       = "WARNING";
220     XML_LOG_LEVEL                        = "TRACE";
221     MME_SCENARIO_PLAYER_LOG_LEVEL        = "TRACE";
222
223     # ASN1 VERBOSITY: none, info, annoying
224     # for S1AP protocol
225     ASN1_VERBOSITY                       = "none";
226 };
227
228 S-GW_LIST_SELECTION = (
229     {ID="tac-lb01.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"
230     ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},
231     {ID="tac-lb02.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"
232     ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},
233     {ID="tac-lb03.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"
234     ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},
235     {ID="tac-lb04.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"
236     ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},
237     {ID="tac-lb05.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"
238     ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},
239 );

```

```
225     {ID="tac-lb06.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
226         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
227     {ID="tac-lb07.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
228         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
229     {ID="tac-lb08.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
230         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
231     {ID="tac-lb09.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
232         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
233     {ID="tac-lb0a.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
234         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
235     {ID="tac-lb0b.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
236         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
237     {ID="tac-lb0c.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
238         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
239     {ID="tac-lb0d.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
240         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
241     {ID="tac-lb0e.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
242         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";},  
243     {ID="tac-lb0f.tac-hb00.tac.epc.mnc093.mcc208.3gppnetwork.org"  
244         ; SGW_IPV4_ADDRESS_FOR_S11="127.0.11.2/8";}  
245 );  
246  
247 };
```

A.4. mme_fd.conf

```

1 # ----- Local -----
2
3 # Uncomment if the framework cannot resolv it.
4 Identity = "epc.5GLaboratory";
5 Realm = "5GLaboratory";
6
7 # TLS configuration (see previous section)
8 TLS_Cred = "/usr/local/etc/oai/freeDiameter/mme.cert.pem",
9           "/usr/local/etc/oai/freeDiameter/mme.key.pem";
10 TLS_CA   = "/usr/local/etc/oai/freeDiameter/mme.cacert.pem";
11
12 # Disable use of TCP protocol (only listen and connect in SCTP)
13 # Default : TCP enabled
14 No_SCTP;
15
16 # This option is ignored if freeDiameter is compiled with DISABLE_SCTP
17   option.
18 # Prefer TCP instead of SCTP for establishing new connections.
19 # This setting may be overwritten per peer in peer configuration blocs
20   .
21 # Default : SCTP is attempted first.
22 Prefer_TCP;
23
24 No_IPv6;
25
26 # Overwrite the number of SCTP streams. This value should be kept low,
27 # especially if you are using TLS over SCTP, because it consumes a lot
28   of
29 # resources in that case. See tickets 19 and 27 for some additional
30   details on
31 # this.
32 # Limit the number of SCTP streams
33 SCTP_streams = 3;
34
35
36 # By default, freeDiameter acts as a Diameter Relay Agent by
37   forwarding all
38 # messages it cannot handle locally. This parameter disables this
39   behavior.
40 NoRelay;
41
42 # Use RFC3588 method for TLS protection, where TLS is negotiated after
43   CER/CEA exchange is completed
44 # on the unsecure connection. The alternative is RFC6733 mechanism,
45   where TLS protects also the
46 # CER/CEA exchange on a dedicated secure port.
47 # This parameter only affects outgoing connections.
48 # The setting can be also defined per-peer (see Peers configuration
49   section).
50 # Default: use RFC6733 method with separate port for TLS.
51
52 #TLS_old_method;
53
54 AppServThreads = 4;
55
56 # Specify the addresses on which to bind the listening server. This
57   must be

```

```
49 # specified if the framework is unable to auto-detect these addresses,
    # or if the
50 # auto-detected values are incorrect. Note that the list of addresses
    # is sent
51 # in CER or CEA message, so one should pay attention to this parameter
    # if some
52 # addresses should be kept hidden.
53 #ListenOn = ;
54
55 Port = 3870;
56 SecPort = 5870;
57
58 # ----- Extensions -----
59
60 # Uncomment (and create rtd.conf) to specify routing table for this
    # peer.
61 #LoadExtension = "rt_default.fdx" : "rtd.conf";
62
63 # Uncomment (and create acl.conf) to allow incoming connections from
    # other peers.
64 #LoadExtension = "acl_wl.fdx" : "acl.conf";
65
66 # Uncomment to display periodic state information
67 #LoadExtension = "dbg_monitor.fdx";
68
69 # Uncomment to enable an interactive Python interpreter session.
70 # (see doc/dbg_interactive.py.sample for more information)
71 #LoadExtension = "dbg_interactive.fdx";
72
73 # Load the RFC4005 dictionary objects
74 #LoadExtension = "dict_nasreq.fdx";
75
76 LoadExtension = "dict_nas_mipv6.fdx";
77 LoadExtension = "dict_s6a.fdx";
78
79 # Load RFC4072 dictionary objects
80 #LoadExtension = "dict_eap.fdx";
81
82 # Load the Diameter EAP server extension (requires diameap.conf)
83 #LoadExtension = "app_diameap.fdx" : "diameap.conf";
84
85 # Load the Accounting Server extension (requires app_acct.conf)
86 #LoadExtension = "app_acct.fdx" : "app_acct.conf";
87
88 # ----- Peers -----
89
90 # The framework will actively attempt to establish and maintain a
    # connection
91 # with the peers listed here.
92 # For only accepting incoming connections, see the acl_wl.fx extension
    # .
93
94 # ConnectPeer
95 # Declare a remote peer to which this peer must maintain a connection.
96 # In addition, this allows specifying non-default parameters for this
    # peer only
97 # (for example disable SCTP with this peer, or use RFC3588-flavour TLS
    # ).
98 # Note that by default, if a peer is not listed as a ConnectPeer entry
    # , an
99 # incoming connection from this peer will be rejected. If you want to
    # accept
```

```
100 # incoming connections from other peers, see the acl_wl.fdx? extension
    which
101 # allows exactly this.
102
103 ConnectPeer= "hss.5GLaboratory" { ConnectTo = "127.0.0.1"; No_SCTP ;
    No_IPv6; Prefer_TCP; No_TLS; port = 3868; realm = "5GLaboratory
    "};};
```


A.5. acl.conf

```
1 # Configuration file for the peer whitelist extension.
2 #
3 # This extension is meant to allow connection from remote peers,
4 #   without actively
5 # maintaining this connection ourselves (as it would be the case by
6 #   declaring the
7 # peer in a ConnectPeer directive).
8 # The format of this file is very simple. It contains a list of peer
9 #   names
10 # separated by spaces or newlines.
11 #
12 # The peer name must be a fqdn. We allow also a special "*" character
13 #   as the
14 # first label of the fqdn, to allow all fqdn with the same domain name
15 #   .
16 # Example: *.example.net will allow host1.example.net and host2.
17 #   example.net
18 #
19 # At the beginning of a line, the following flags are allowed (case
20 #   sensitive) -- either or both can appear:
21 # ALLOW_OLD_TLS : we accept unprotected CER/CEA exchange with Inband-
22 #   Security-Id = TLS
23 # ALLOW_IPSEC   : we accept implicitly protected connection with with
24 #   peer (Inband-Security-Id = IPSec)
25 # It is specified for example as:
26 # ALLOW_IPSEC vpn.example.net vpn2.example.net *.vpn.example.net
27
28 ALLOW_OLD_TLS    *.5GLaboratory
```

```
#####
# Licensed to the OpenAirInterface (OAI) Software Alliance under one
# or more
# contributor license agreements. See the NOTICE file distributed
# with
# this work for additional information regarding copyright ownership.
# The OpenAirInterface Software Alliance licenses this file to You
# under
# the Apache License, Version 2.0 (the "License"); you may not use
# this file
# except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#-----
# For more information about the OpenAirInterface (OAI) Software
# Alliance:
# contact@openairinterface.org
#####
S-GW :
{
    NETWORK_INTERFACES :
    {
        # S-GW binded interface for S11 communication (GTPV2-C), if
        # none selected the ITTI message interface is used
        SGW_INTERFACE_NAME_FOR_S11 = "lo";
        # STRING, interface name, YOUR
        NETWORK CONFIG HERE
        SGW_IPV4_ADDRESS_FOR_S11 = "127.0.11.2/8";
        # STRING, CIDR, YOUR NETWORK CONFIG HERE

        # S-GW binded interface for S1-U communication (GTPV1-U) can
        # be ethernet interface, virtual ethernet interface, we don't
        # advise wireless interfaces
        SGW_INTERFACE_NAME_FOR_S1U_S12_S4_UP = "eth1";
        # STRING, interface name, YOUR
        NETWORK CONFIG HERE, USE "lo" if S-GW run on eNB host
        SGW_IPV4_ADDRESS_FOR_S1U_S12_S4_UP = "192.168.1.1/24";
        # STRING, CIDR, YOUR NETWORK CONFIG HERE
        SGW_IPV4_PORT_FOR_S1U_S12_S4_UP = 2152;
        # INTEGER, port number, PREFER NOT
        CHANGE UNLESS YOU KNOW WHAT YOU ARE DOING

        # S-GW binded interface for S5 or S8 communication, not
        # implemented, so leave it to none
        SGW_INTERFACE_NAME_FOR_S5_S8_UP = "none";
        # STRING, interface name, DO NOT
        CHANGE (NOT IMPLEMENTED YET)
        SGW_IPV4_ADDRESS_FOR_S5_S8_UP = "0.0.0.0/24";
        # STRING, CIDR, DO NOT CHANGE (NOT
        IMPLEMENTED YET)
    }
}
```

```

37     };
38
39     INTERTASK_INTERFACE :
40     {
41         # max queue size per task
42         ITTI_QUEUE_SIZE          = 2000000;
43                                     # INTEGER
44     };
45
46     LOGGING :
47     {
48         # OUTPUT choice in { "CONSOLE", "SYSLOG", 'path to file', "
49         #   IPv4@': 'TCP port num' }
50         # 'path to file' must start with '.' or '/'
51         # if TCP stream choice, then you can easily dump the traffic
52         #   on the remote or local host: nc -l 'TCP port num' >
53         #   received.txt
54         OUTPUT                    = "CONSOLE";
55                                     # see 3 lines
56         # above
57         #OUTPUT                   = "SYSLOG";
58                                     # see 4 lines
59         # above
60         #OUTPUT                   = "/tmp/spgw.log";
61                                     # see 5 lines above
62         #OUTPUT                   = "127.0.0.1:5656";
63                                     # see 6 lines above
64
65         # THREAD_SAFE choice in { "yes", "no" } means use of thread
66         #   safe intermediate buffer then a single thread pick each
67         #   message log one
68         # by one to flush it to the chosen output
69         THREAD_SAFE               = "no";
70
71         # COLOR choice in { "yes", "no" } means use of ANSI styling
72         #   codes or no
73         COLOR                     = "yes";
74
75         # Log level choice in { "EMERGENCY", "ALERT", "CRITICAL", "
76         #   ERROR", "WARNING", "NOTICE", "INFO", "DEBUG", "TRACE" }
77         ASYNC_SYSTEM              = "TRACE";
78         UDP_LOG_LEVEL             = "TRACE";
79         GTPV1U_LOG_LEVEL          = "TRACE";
80         GTPV2C_LOG_LEVEL          = "TRACE";
81         SPGW_APP_LOG_LEVEL        = "TRACE";
82         S11_LOG_LEVEL             = "TRACE";
83         UTIL_LOG_LEVEL            = "TRACE";
84         ITTI_LOG_LEVEL            = "WARNING";
85     };
86 };
87
88 P-GW =
89 {
90     NETWORK_INTERFACES :
91     {
92         # P-GW binded interface for S5 or S8 communication, not
93         #   implemented, so leave it to none
94         PGW_INTERFACE_NAME_FOR_S5_S8 = "none";
95                                     # STRING, interface name, DO NOT
96                                     # CHANGE (NOT IMPLEMENTED YET)
97     };
98 };

```

```

81     # P-GW binded interface for SGI (egress/ingress internet
      traffic)
82     PGW_INTERFACE_NAME_FOR_SGI          = "eth0";
      # STRING, YOUR NETWORK CONFIG HERE
83     PGW_MASQUERADE_SGI                  = "no";
      # STRING, {"yes", "no"}. YOUR
      NETWORK CONFIG HERE, will do NAT for you if you put "yes".
84     UE_TCP_MSS_CLAMPING                  = "no";
      # STRING, {"yes", "no"}.
85 };
86
87 # Pool of UE assigned IP addresses
88 # Do not make IP pools overlap
89 # first IPv4 address X.Y.Z.1 is reserved for GTP network device on
      SPGW
90 # Normally no more than 16 pools allowed, but since recent GTP
      kernel module use, only one pool allowed (TODO).
91 IP_ADDRESS_POOL :
92 {
93     IPV4_LIST = (
94         "172.16.0.0/12"
85                                     #
86                                     STRING, CIDR, YOUR NETWORK CONFIG HERE.
95     );
96 };
97
98 # DNS address communicated to UEs
99 DEFAULT_DNS_IPV4_ADDRESS      = "8.8.8.8";
      # YOUR NETWORK CONFIG HERE
100 DEFAULT_DNS_SEC_IPV4_ADDRESS = "8.8.4.4";
      # YOUR NETWORK CONFIG HERE
101
102 # Non standard feature, normally should be set to "no", but you
      may need to set to yes for UE that do not explicitly request a
      PDN address through NAS signalling
103 FORCE_PUSH_PROTOCOL_CONFIGURATION_OPTIONS = "no";
      # STRING, {"yes", "no"}.
104 UE_MTU                                = 1500
      # INTEGER
105 GTPV1U_REALIZATION                = "GTP_KERNEL_MODULE";
      # STRING {"NO_GTP_KERNEL_AVAILABLE", "
      GTP_KERNEL_MODULE", "GTP_KERNEL"}. In a container you may not
      be able to unload/load kernel modules.
106
107 PCEF :
108 {
109     PCEF_ENABLED                    = "yes";
      # STRING, {"yes", "no"}, if yes
      then all parameters bellow will/should be taken into account
110     TRAFFIC_SHAPPING_ENABLED        = "yes";
      # STRING, {"yes", "no"}, TODO,
      should finally work for egress but only on ingress bearers
      and not on ingress SDF flows
111     TCP_ECN_ENABLED                  = "yes";
      # STRING, {"yes", "no"}, TCP
      explicit congestion notification
112     AUTOMATIC_PUSH_DEDICATED_BEARER_PCC_RULE= 0;
      # INTEGER [ 0..n], SDF
      identifier (Please check with enum sdf_id_t in
      pgw_pcef_emulation.h,
113 # !!!!!!!!!!!!! BE CAREFULL, EXPERIMENTAL !!!!!!!!!!!!! may need to
      be updated, and some are not available )

```

```

114 # 0 = No push of dedicated bearer
115 # 17 = SDF_ID_GBR_VOLTE_16K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
116 # 18 = SDF_ID_GBR_VOLTE_24K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
117 # 19 = SDF_ID_GBR_VOLTE_40K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
118 # 20 = SDF_ID_GBR_VOLTE_64K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
119 # 21 = SDF_ID_GBR_VILTE_192K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
120 # 22 = SDF_ID_GBR_VILTE_384K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
121 # 23 = SDF_ID_GBR_VILTE_768K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
122 # 24 = SDF_ID_GBR_VILTE_2M, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
123 # 25 = SDF_ID_GBR_VILTE_4M, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
124 # 26 = SDF_ID_GBR_NON_CONVERSATIONAL_VIDEO_256K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
125 # 27 = SDF_ID_GBR_NON_CONVERSATIONAL_VIDEO_512K, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
126 # 28 = SDF_ID_GBR_NON_CONVERSATIONAL_VIDEO_1M, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
127 # 29 = SDF_ID_NGBR_IMS_SIGNALLING, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
128 # 30 = SDF_ID_NGBR_DEFAULT_PREMIUM, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
129 # 31 = SDF_ID_NGBR_DEFAULT, // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
130 # 32 = SDF_ID_TEST_PING // see corresponding
      TFT and QOS params in pgw_pcef_emulation.h
131
132     DEFAULT_BEARER_STATIC_PCC_RULE = 31; // SDF identifier for
      default bearer
133     PUSH_STATIC_PCC_RULES = (31); // List of SDF
      identifiers
134
135     # Waiting for HSS APN-AMBR IE ...
136     APN_AMBR_UL = 500000;
      # Maximum UL bandwidth that can be
      used by non guaranteed bit rate traffic in Kbits/seconds.
137     APN_AMBR_DL = 500000;
      # Maximum DL bandwidth that can be
      used by non guaranteed bit rate traffic in Kbits/seconds.
138 };
139 };

```


Apéndice B

Archivos de configuración de OAISIM

B.1. enb.band7.tm1.usrpb210.conf

```
1 Active_eNBs = ( "eNB_Eurecom_LTEBox");
2 # Asn1_verbosity, choice in: none, info, annoying
3 Asn1_verbosity = "none";
4
5 eNBs =
6 (
7 {
8     ////////// Identification parameters:
9     eNB_ID      = 0xe00;
10
11     cell_type   = "CELL_MACRO_ENB";
12
13     eNB_name    = "eNB_Eurecom_LTEBox";
14
15     // Tracking area code, 0x0000 and 0xfffe are reserved values
16     tracking_area_code = "1";
17
18     mobile_country_code = "208";
19
20     mobile_network_code = "94";
21
22     ////////// Physical parameters:
23
24     component_carriers = (
25     {
26         frame_type           = "FDD";
27         tdd_config            = 3;
28         tdd_config_s         = 0;
29         prefix_type          = "
                NORMAL";
30         eutra_band            = 7;
31         downlink_frequency   =
                2680000000L;
```

```

32      uplink_frequency_offset                =
        -120000000;
33      Nid_cell                              = 0;
34      N_RB_DL                               = 25;
35      Nid_cell_mbsfn                        = 0;
36      nb_antenna_ports                      = 1;
37      nb_antennas_tx                        = 1;
38      nb_antennas_rx                        = 1;
39      tx_gain                               = 90;
40      rx_gain                               = 125;
41      prach_root                            = 0;
42      prach_config_index                    = 0;
43      prach_high_speed                      = "
        DISABLE";
44      prach_zero_correlation                = 1;
45      prach_freq_offset                     = 2;
46      pucch_delta_shift                     = 1;
47      pucch_nRB_CQI                         = 1;
48      pucch_nCS_AN                          = 0;
49      pucch_n1_AN                           = 32;
50      pdsch_referenceSignalPower            = -29;
51      pdsch_p_b                             = 0;
52      pusch_n_SB                           = 1;
53      pusch_enable64QAM                     = "
        DISABLE";
54      pusch_hoppingMode                     = "
        interSubFrame";
55      pusch_hoppingOffset                   = 0;
56      pusch_groupHoppingEnabled             = "
        ENABLE";
57      pusch_groupAssignment                 = 0;
58      pusch_sequenceHoppingEnabled         = "
        DISABLE";
59      pusch_nDMRS1                          = 1;
60      phich_duration                        = "NORMAL";
61      phich_resource                         = "ONESIXTH
        ";
62      srs_enable                            = "DISABLE
        ";
63      /* srs_BandwidthConfig                 =;
64      srs_SubframeConfig                     =;
65      srs_ackNackST                          =;
66      srs_MaxUpPts                           =;*/
67
68      pusch_p0_Nominal                      = -90;
69      pusch_alpha                           = "AL1";
70      pucch_p0_Nominal                      = -96;
71      msg3_delta_Preamble                   = 6;
72      pucch_deltaF_Format1                  = "deltaF2";
73      pucch_deltaF_Format1b                 = "deltaF3";
74      pucch_deltaF_Format2                  = "deltaF0";
75      pucch_deltaF_Format2a                 = "deltaF0";
76      pucch_deltaF_Format2b                 = "deltaF0";
77
78      rach_numberOfRA_Preambles              = 64;
79      rach_preamblesGroupAConfig            = "DISABLE
        ";
80      /*
81      rach_sizeOfRA_PreamblesGroupA          = ;
82      rach_messageSizeGroupA                 = ;
83      rach_messagePowerOffsetGroupB          = ;
84      */

```



```

85     rach_powerRampingStep                = 4;
86     rach_preambleInitialReceivedTargetPower = -108;
87     rach_preambleTransMax                = 10;
88     rach_raResponseWindowSize            = 10;
89     rach_macContentionResolutionTimer    = 48;
90     rach_maxHARQ_Msg3Tx                  = 4;
91
92     pcch_default_PagingCycle              = 128;
93     pcch_nB                              = "oneT";
94     bcch_modificationPeriodCoeff          = 2;
95     ue_TimersAndConstants_t300            = 1000;
96     ue_TimersAndConstants_t301            = 1000;
97     ue_TimersAndConstants_t310            = 1000;
98     ue_TimersAndConstants_t311            = 10000;
99     ue_TimersAndConstants_n310            = 20;
100    ue_TimersAndConstants_n311            = 1;
101
102    ue_TransmissionMode                    = 1;
103 }
104 );
105
106 srb1_parameters :
107 {
108     # timer_poll_retransmit = (ms) [5, 10, 15, 20,... 250, 300,
109         350, ... 500]
110     timer_poll_retransmit          = 80;
111
112     # timer_reordering = (ms) [0,5, ... 100, 110, 120, ... ,200]
113     timer_reordering               = 35;
114
115     # timer_reordering = (ms) [0,5, ... 250, 300, 350, ... ,500]
116     timer_status_prohibit         = 0;
117
118     # poll_pdu = [4, 8, 16, 32 , 64, 128, 256, infinity(>10000)]
119     poll_pdu                       = 4;
120
121     # poll_byte = (kB)
122         [25,50,75,100,125,250,375,500,750,1000,1250,1500,2000,3000,
123         infinity(>10000)]
124     poll_byte                      = 99999;
125
126     # max_retx_threshold = [1, 2, 3, 4 , 6, 8, 16, 32]
127     max_retx_threshold             = 4;
128 }
129
130 # ----- SCTP definitions
131 Sctp :
132 {
133     # Number of streams to use in input/output
134     SCTP_INSTREAMS = 2;
135     SCTP_OUTSTREAMS = 2;
136 };
137
138 ////////////// MME parameters:
139 mme_ip_address = ( { ipv4          = "192.168.12.70";
140                     ipv6          = "192:168:30::17";
141                     active        = "yes";
142                     preference    = "ipv4";
143                     }
144 );
145
146 NETWORK_INTERFACES :

```

```

144 {
145     ENB_INTERFACE_NAME_FOR_S1_MME           = "eth0";
146     ENB_IPV4_ADDRESS_FOR_S1_MME            =
147         "192.168.12.150/24";
148
149     ENB_INTERFACE_NAME_FOR_S1U              = "eth0";
150     ENB_IPV4_ADDRESS_FOR_S1U                =
151         "192.168.12.150/24";
152     ENB_PORT_FOR_S1U                        = 2152; # Spec 2152
153 };
154
155 log_config :
156 {
157     global_log_level                        = "info";
158     global_log_verbosity                    = "medium";
159     hw_log_level                           = "info";
160     hw_log_verbosity                       = "medium";
161     phy_log_level                          = "info";
162     phy_log_verbosity                      = "medium";
163     mac_log_level                         = "info";
164     mac_log_verbosity                     = "high";
165     rlc_log_level                         = "info";
166     rlc_log_verbosity                     = "medium";
167     pdcp_log_level                       = "info";
168     pdcp_log_verbosity                   = "medium";
169     rrc_log_level                        = "info";
170     rrc_log_verbosity                   = "medium";
171 };
172 }
173 );

```