ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

DESPLIEGUE Y EVALUACIÓN DE UNA RED CELULAR LTE MEDIANTE SOLUCIONES SDR

(Deployment and evaluation of a SDR-based LTE network)

Para acceder al Titulo de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Alberto Castilla Gómez

Julio - 2020

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Alberto Castilla Gómez
Directores: Luis Francisco Diez Fernández y Ramón Agüero Calvo
Título: "Despliegue y evaluación de una red celular LTE mediante soluciones SDR"
Title: "Deployment and evaluation of a SDR-based LTE network"

Presentado a examen el día: 15 de julio de 2020

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): García Gutiérrez, Alberto Eloy
Secretario (Apellidos, Nombre): Diez Fernández, Luis Francisco
Vocal (Apellidos, Nombre): Fuentes Saez, Pablo

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

Agradecimientos

A Ramón Agüero y a Luis Francisco Diez por su profesionalidad y dedicación, sobre todo en los últimos meses vividos debido a la crisis sanitaria.

A mis compañeros de clase, con especial mención a Laura, Cristina, Sergio y Christian, por su apoyo y empatía.

Por último pero tal vez los más importantes, a mis padres y mi hermano: pilares fundamentales de mi vida.

Resumen

Los entornos de simulación de redes juegan un papel fundamental en el desarrollo de las mismas. Su importancia se justifica en su capacidad de anticiparse a potenciales problemas u obstáculos que puedan surgir a la hora de realizar el despliegue de la infraestructura diseñada, permitiendo un ahorro en términos económicos y temporales.

En determinados supuestos, estas herramientas no aportan el grado de veracidad demandado. Asimismo, la utilización de implementaciones reales para realizar experimentos conlleva la necesidad de manejar una electrónica muy especializada, lo que implica un incremento de los costes.

Con el propósito de reducir estos inconvenientes, en los últimos años se están desarrollando implementaciones software de diferentes tecnologías de radiocomunicaciones (Software Defined Radio), que empleando electrónica más sencilla, permiten desplegar entornos realistas y verosímiles.

A lo largo del proyecto, se ha aplicado la tecnología *srsLTE*, desarrollada por Software Radio Systems (SRS), que permite desplegar una red Long Term Evolution (LTE).

Una vez desplegada, se han analizado los requisitos computacionales necesarios para su despliegue y todos los imprevistos que se han encontrado. Posteriormente, se ha estudiado su rendimiento y fiabilidad.

La evaluación de la red se ha realizado en dos entornos: en primer lugar, mediante la herramienta IPERF, se ha caracterizado su comportamiento empleando tráfico sintético y, posteriormente, se ha verificado su viabilidad en un escenario con tráfico real, mediante el despliegue de un servidor de streaming de video utilizando Video LAN Client (VLC).

Abstract

Network simulator environments have a fundamental part in the development of network. Its importance is justified in its ability to anticipate to potencial problems which can appear during the deployment of the designed infrastructure, allowing a saving in economics and temporal terms.

In some cases, these tools do not provide the expected degree of accuracy. Moreover, the usage of real deployments to make experiments implies to operate with a specialized electronic which increases the project's costs.

In order to reduce these problems, it has been developed software deployments of radiocommunication technologies (SDR) which uses a simpler electronic and it allows to develop plausible environments.

In this project, it has been applied *srsLTE*, which was developed by SRS, that allows to deploy a LTE network.

After the network deployment, it has been studied the needed computational requirements and the founded contingencies. Afterwards, it has been examined its performance and reability.

The network assessment has been undertaken in two environments: firstly, it has been characterized the network performance using sintetic traffic generated by IPERF. Subsequentely, it has been deployed a video streaming server using VLC in order to study the feasibily of the network in real conditions.

Índice de figuras

Índice de tablas

Lista de acrónimos

3GPP 3G Partnership Project. 15, 16, 19, 24, 38, 59

5GNR 5G New Radio. 16

ADC Analog Digital Converter. 21, 28

AMPS Advanced Mobile Phone System. 15

AR Augmented Reality. 17

BBU BaseBand Unit. 19, 20

BLER Block Error Rate. 30

BW Bandwidth. 6, 19, 28, 38, 60, 61

CF Core Framework. 22

CORBA Common Object Request Broker Architecture. 22

COTS Commercial Off-The-Shelf. 25, 26

DAC Digital Analog Converter. 21, 28

DL Downlink. 30, 33, 38, 58

DSP Digital Signal Processor. 26

EARFCN E-UTRA Absolute Radio Frequency Channel Number. 6, 38, 39

EB Estación Base. 17, 23, 26, 32–34, 39, 40, 42, 44, 46

EDGE Enhanced Data GSM Enviroment. 15

ENB ENodeB. 4, 6, 29, 31–36, 38, 42, 44, 45, 49, 58

EPC Evolved Packet Core. 29, 31, 34, 36

FCI Framework Control Interfaces. 22

FDMA Frecuency Division Multiple Access. 15

FO Fibra Óptica. 19

FPGA Field-Programmable Gate Array. 26–28, 30, 56

FR Frequency Range. 16

FSI Framework Service Interfaces. 22

GOPS Giga Operations Per Second. 25

GPP General Purpose Processor. 26

GPRS General Packet Radio Service. 15

GPU Graphics Processing Unit. 26

GSM Global System for Mobile Communications. 15

GTP GPRS Tunnelling Protocol. 34, 35

GTP-U GPRS Tunnelling Protocol User Plane. 34

GW Gateway. 34

HARQ Hybrid Automatic Repeat Request. 34

HSPA High Speed Packet Access. 16

HSS Home Subscriber Service. 35, 36

IoT Internet of Things. 13, 17, 20, 25

IP Internet Protocol. 15, 16, 32–36, 49

LNA Low Noise Amplifier. 21

LTE Long Term Evolution. 2, 5, 13, 14, 16, 24, 25, 29, 40, 42, 60, 62

LTS Long Term Support. 29, 56

M2M Machine to Machine. 25

MAC Media Access Control. 34, 57

MBMS Multimedia Broadcast Multicast Service. 36

MIMO Multiple Input Multiple Output. 17, 28

MME Mobility Managment Entity. 35, 36

NAS Non-Access Stratum. 34, 57

NFV Network Function Virtualization. 18

OAI Open Air Interface. 24

OFDMA Orthogonal Frecuency Division Multiple Access. 16

OMG Object Management Group. 22

PDCP Packet Data Convergence Protocol. 34

PDU Packet Data Unit. 34

PGW Packet Gateway. 34–36

PLMN Public Land Mobile Network. 34, 40

PRB Physical Resource Block. 6, 12, 36, 39, 40, 42–46, 49, 51, 52, 60, 61

PtP Point to Point. 32

QoS Quality of Service. 17, 35

RAN Radio Access Network. 18

RE Resource Element. 60

RLC Radio Link Control. 34

ROHC Robust Header Compression. 34

RRC Radio Resource Control. 34, 37, 40

RRH Remote Radio Head. 19

RRU Remote Radio Unit. 19, 20

S1-AP S1 Application Protocol. 34

SA Stand-Alone. 17

SCA Sofware Communications Architecture. 22

SDR Software Defined Radio. 4, 13, 14, 20, 22–25, 27

SGW Servicing Gateway. 35, 36

SIB System Information Block. 36

SIMD Single Instruction Multiple Data. 25

SMS Short Message Service. 15

SO Sistema Operativo. 27, 29, 30, 55

SRS Software Radio Systems. 13, 24, 55

TACS Total Access Communication System. 15

TB Transport Block. 34

TCP Transport Control Protocol. 4–6, 41, 45, 46, 62, 63

TDMA Time Division Multiple Access. 15

UDP User Datagram Protocol. 4–6, 41, 45, 47, 48, 61–64

UE User Equipment. 4, 6, 29–34, 36, 38–42, 44, 45, 58, 59

UHD USRP Hardware Driver. 27, 30, 56

UL Uplink. 38, 58

USRP Universal Software Radio Pherifierical. 25–27, 30

V2I Vehicle to Infrastructure. 17

V2P Vehicle to Pedestrians. 17

V2V Vehicle to Vehicle. 13, 17, 20

VLC Video LAN Client. 14, 41

VoD Video On Demand. 16

VoIP Voice Over IP. 16

VPN Virtual Private Network. 16

VR Virtual Reality. 17

WCDMA Wideband Code Division Multiple Access. 15

XML Extensible Markup Language. 22

Índice general

Capítulo 1

Introducción

El Software Defined Radio (SDR) se ha convertido en una tecnología vanguardista en los últimos años. A pesar de que su origen se remonta a finales del siglo pasado (Joseph Mitola III, 1993), su desarrollo se está produciendo durante los últimos años.

Su gran impacto se justifica en su capacidad para desplegar diferentes protocolos de radiocomunicaciones sin la necesidad de cambiar el hardware y, simplemente, precisando una actualización o modificaciones en el software. Gracias a esto se consigue garantizar la interoperabilidad de un mismo hardware para trabajar en diferentes escenarios. Por ejemplo, empleando el mismo dispositivo electrónico se puede desplegar una red WiFi (que opere en la banda de 2.4 GHz) o implementar una red celular de 2ª Generación que trabaje en la banda de 900 MHz.

Esta flexibilidad de la tecnología permite un ahorro económico importante para las empresas, ya que, no es preciso adquirir un hardware específico para cada sistema de radiocomunicaciones que se desee implementar.

Además, SDR es una técnica empleada en el ámbito militar, en las comunicaciones espaciales, las comunicaciones entres V2V y el Internet of Things (IoT).

En este trabajo se ha desplegado una red celular LTE mediante el uso de *srsLTE* (desarrollado por la empresa irlandesa Software Radio Systems (SRS)) con el objetivo de establecer una plataforma para evaluar el comportamiento de dicha tecnología en varios escenarios.

1.1. Objetivos del proyecto

Como ya se ha comentado previamente, el proyecto ha sido desarrollado gracias al uso de la tecnología, *srsLTE*, desarrollada por SRS. Esta ha permitido desplegar una red LTE con los siguientes propósitos:

- Análisis de los requisitos computacionales necesarios para su implementación.
- Estudio de los percances encontrados en la instalación y configuración de la red.

- Evaluación de la tecnología LTE mediante el uso de tráfico sintético mediante el uso de la herramienta IPERF.
- Despliegue de un servidor de streaming de video en la red LTE con el objetivo de analizar y estudiar su funcionamiento bajo tráfico real. Para ello, se empleará el software VLC.

1.2. Organización de la memoria

Este documento se ha dividido en 5 capítulos. Tras la introducción realizada en la anterior sección, donde se exponen los objetivos del proyecto, el resto del texto sigue la siguiente estructura:

En el Capítulo 2 se expone la evolución de las redes móviles para contextualizar la situación actual. Se describe 5G y, por último, se explica qué es el SDR. Además, respecto a SDR, se detalla su filosofía y evolución, las principales alternativas (software y hardware) que hay en el mercado y se exponen las razones fundamentales que han permitido hecho a favor de una de las alternativas.

El tercer capítulo resume el proceso realizado a lo largo del proyecto. En primer lugar, se expone el procedimiento de instalación del software srsLTE, describiéndose todos los pasos efectuados y las limitaciones encontradas. Posteriormente, se describe la red que se ha desplegado en la instalación, se exponen los grados de libertad que se tienen para poder manipularla y todos los problemas que se han ido encontrando. Tras ello, se realiza una breve argumentación de las modificaciones que se han efectuado sobre el código srsLTE con el objetivo de adecuar el sistema a las pruebas que se realizaron. Posteriormente, se explica el procedimiento desarrollado para poder automatizar las pruebas con tráfico sintético. Además, se pormenoriza el proceso de parseo que se ha empleado para adecuar los ficheros de resultados, obtenidos en los experimentos llevados a cabo, para poder realizar su análisis. Por último, se explica el proceso utilizado para la implementación de un servidor VLC que permite realizar pruebas con tráfico real.

El Capítulo 4 detalla los diferentes escenarios en los que se han practicado las pruebas y proporciona una exposición detallada de los resultados obtenidos.

El último capítulo se usa para resaltar los resultados conseguidos y exponer las conclusiones obtenidas. En último término, se comentan las posibles líneas de trabajo futuras que puedan surgir del proyecto.

Capítulo 2

Introducción teórica

2.1. Evolución de las redes celulares

A continuación, se resumen las principales características de cada generación de las redes móviles con el objetivo de contextualizar al lector la tecnología 5G [1] [2] [3]:

- **Primera Generación (1G)**: Se trata de un sistema de comunicaciones móviles analógico que surge en los años 80. Emplea Frecuency Division Multiple Access (FDMA) como mecanismo de acceso al medio, y cada canal tiene un ancho de 30 kHz. En España se empleó el sistema Total Access Communication System (TACS), que operaba en la banda de 900 MHz, y en Estados Unidos el sistema AMPS (800 MHz).

  Su uso se limitaba a las llamadas telefónicas. Entre las principales carencias de 1G destaca que no se cifraban las llamadas telefónicas, la baja calidad de las mismas y su escasa velocidad de transmisión (inferior a 9.6 kbps).

- **Segunda Generación (2G)**: En los años 90, se desarrolla el primer sistema de comunicaciones móviles digital. Basado en Time Division Multiple Access (TDMA), emplea el estándar Global System for Mobile Communications (GSM) en Europa y trabaja en las bandas de 900 y 1800 MHz, con portadoras de 200 kHz.

  Respecto a la generación anterior, mejora la calidad de la voz, cifra las comunicaciones, permite el roaming internacional, la identificación de llamadas y el envío de Short Message Service (SMS).

  Posteriormente, en el año 2000 se introduce el estándar General Packet Radio Service (GPRS), que permitía la transmisión de datos a una tasa de 115 kbps y, después, mediante el uso de Enhanced Data GSM Enviroment (EDGE) se alcanzan velocidades de 384 kbps.

- **Tercera Generación (3G)**: En los años 2000, el foro 3G Partnership Project (3GPP) crea el estándar IMT-2000, usado para estandarizar la estructura de red. Se basa en Wideband Code Division Multiple Access (WCDMA) como mecanismo de acceso al medio, la conmutación de paquetes y el protocolo Internet Protocol (IP).

Opera en las bandas de 900 MHz y 2.1 GHz, con portadoras de 5 MHz posibilitando alcanzar velocidades de hasta 2 Mbps.

A nivel técnico, 3G supone una mejora importante en la calidad de las comunicaciones, un incremento de la velocidad y la posibilidad de dar servicio a una mayor cantidad de usuarios. Estos progresos permiten que surgan nuevos servicios como videollamadas, Video On Demand (VoD), aplicaciones de mensajería instantánea, Virtual Private Network (VPN) móviles y la geolocalización. Con motivo de estos nuevos servicios aparecen los smartphones, convirtiendo a los terminales en instrumentos cuyo único fin no son las llamadas telefónicas.

El estándar High Speed Packet Access (HSPA) permitió optimizar el uso de los recursos móviles consiguiéndose aumentar la tasa binaria ofrecida a los usuarios (hasta 10 Mbps).

- **Cuarta Generación (4G)**: A partir de 2010, comienzan los despliegues de las primeras redes que utilizan el estándar LTE. Tiene como objetivo garantizar una latencia baja (inferior a los 10 ms), una velocidad elevada (alcanzándose más de 150 Mbps en descarga) y una reducción de costes. LTE opera en las bandas de 800 MHz, 1.8 y 2.6 GHz, y garantiza el acceso al medio mediante el uso de Orthogonal Frecuency Division Multiple Access (OFDMA) con portadoras de 180 kHz.

  Se trata de una red IP y la comunicación está orientada a paquetes, por lo que esta tecnología no está pensada para cursar llamadas. Además, se impide la posibilidad de realizar handovers hacia células de generaciones anteriores, por lo que, si se realizase una llamada, esta se perdería al entrar en una zona donde 4G no tenga cobertura.

  En conclusión, esta generación está enfocada en servicios como Voice Over IP (VoIP), streamings de video o TV Móvil.

2.2. La Quinta Generación: 5G

La Quinta Generación de redes móviles, conocida comúnmente como 5G, está siendo estandarizada por el grupo 3GPP. Su principal propósito es conectar dispositivos y máquinas, y no solo personas.

Respecto a 4G, tiene como objetivo reducir la latencia hasta 1-5 ms, garantizar tasas superiores a los 10 Gbps y conseguir conectar de forma simultánea hasta 100 veces más dispositivos. Por último, tiene como motivación principal garantizar una cobertura total, una altísima disponibilidad y reducir el consumo energético de la red en un 90 %.

Al igual que LTE, 5G se basa en el uso de OFDMA como mecanismo para acceder al medio. Además, emplea 5G New Radio (5GNR) para dividir el espectro en 2 rangos:

- **Frequency Range 1 (FR1)**: Frecuencias entre 450 MHz y 6 GHz. Ofrece una mayor cobertura y penetración en edificios.
- **Frequency Range 2 (FR2)**: Abarca desde los 24.25 GHz hasta los 52.6 GHz, se caracteriza por tener menor cobertura que FR1 pero permite realizar transferencias

a velocidades mucho mayores. Esto provoca que estas redes, conocidas como 5G Stand-Alone (SA), estén formadas por células más pequeñas y, por tanto, origine una mayor densificación y un incremento de los costes de despliegue.

La técnica Multiple Input Multiple Output (MIMO) consiste en el uso de múltiples antenas, tanto en transmisión como en recepción, para aprovechar la propagación multicamino aumentando la eficiencia espectral de la comunicación y obteniendo como resultado una mayor tasa binaria y un incremento de la fiabilidad. Gracias al uso de esta técnica se consigue aumentar la cantidad de usuarios que se puedan conectar de forma simultánea a una misma Estación Base (EB).

También es importante destacar sobre 5G su capacidad de adaptación a las necesidades de los usuarios y dispositivos.

Por ejemplo, un sensor de un aparcamiento que utilice tecnología 5G necesita que su consumo energético sea bajo. Sin embargo, los sensores de los vehículos autónomos demandan una latencia muy baja con el fin de reaccionar ante cualquier imprevisto, pero no tienen tantas limitaciones energéticas.

Esa capacidad de adaptación se consigue gracias a la inyección de funciones de inteligencia en la red como es el *Network Slicing* y la virtualización. El *Network Slicing* permite disponer de múltiples redes lógicas virtuales e independientes denominadas *slices* sobre una misma estructura física [4].

Abarca una gran cantidad de partes de la red: la red de acceso, la red de transporte y la red troncal. Es importante destacar que se puede desplegar por múltiples operadores compartiendo la red física y separando a sus clientes en *slices*.

Cada *slice* se define de forma independiente, adaptando su configuración (Quality of Service (QoS), latencia, seguridad, movilidad, eficiencia energética, ...) para los servicios que va a ofrecer (IoT, servicios de emergencia, Virtual Reality (VR)/Augmented Reality (AR), vehículos autónomos y sus comunicaciones V2V, Vehicle to Infrastructure (V2I) y Vehicle to Pedestrians (V2P)).

Las Figuras 2.1 y 2.2 representan redes 5G donde la principal diferencia entre ellas reside en que la segunda emplea *Network Slicing*:

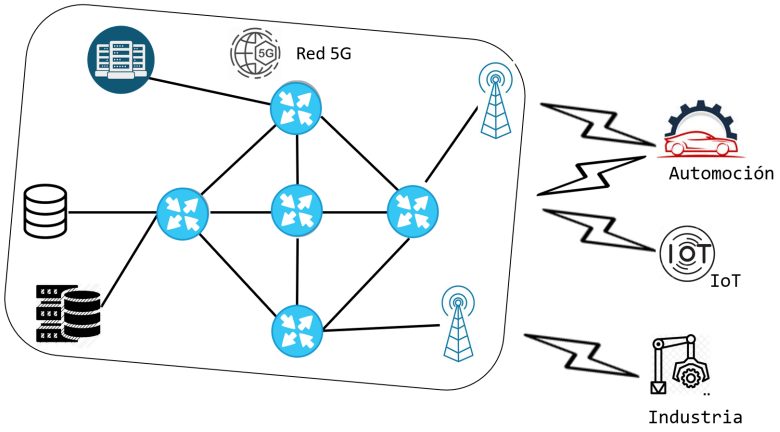

Figura 2.1: Red 5G

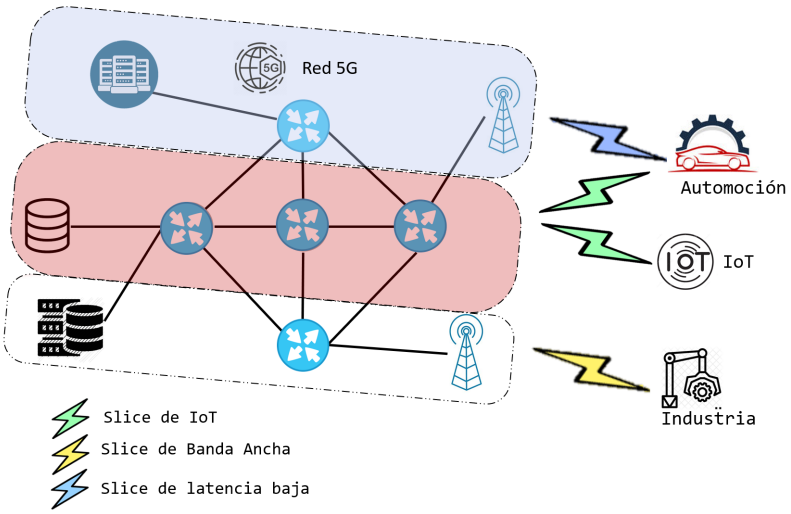

Figura 2.2: Network Slicing 5G

Este hecho aporta robustez a la red y una mayor seguridad, ya que, si algún slice sufre algún imprevisto (caída, sobrecarga o ataque) solamente se verá afectado él y no el resto.

La Figura 2.3 muestra los principales parámetros de configuración de cada *slice*:

La *virtualización de funciones de red* es la otra técnica que se emplea en 5G para introducir inteligencia en la misma. El Network Function Virtualization (NFV) consiste en alejar funciones de la red de acceso a la red troncal, donde se procesarán de forma centralizada. De este modo, se consigue que determinadas tareas, que hasta ahora cumplían dispositivos hardware, puedan ser realizadas por servidores virtuales. Este hecho tiene una repercusión económica importante: reducción del número de dispositivos físicos, reducción de costes y mantenimiento.

Es importante considerar que, a la hora de desplegar redes NFV operacionales, es imprescindible garantizar una coordinación entre la virtualización, orquestación y la automatización de los servicios virtualizados.

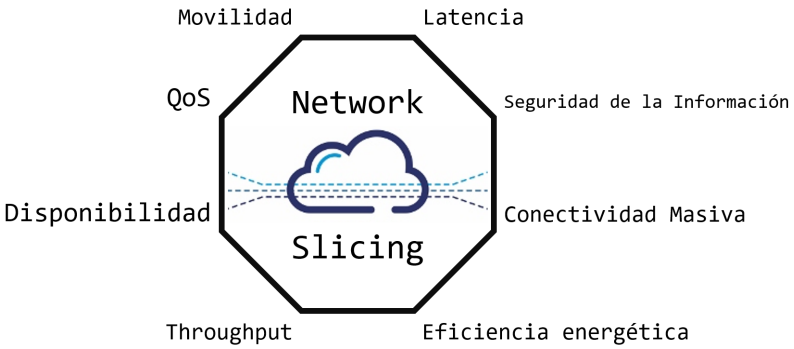

Figura 2.3: Configuración de parámetros de un slice

Con la finalidad de desarrollar, estandarizar y extender el NFV en las Radio Access Network (RAN) surge el proyecto xRAN. La Figura 2.4 resume las principales empresas que forman parte de este proyecto:

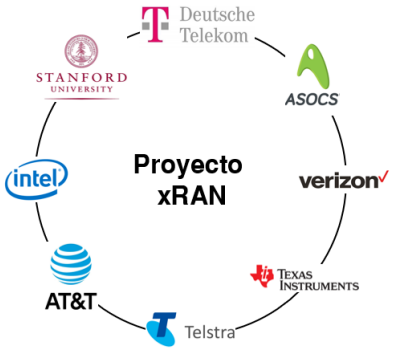

Figura 2.4: Principales entidades que conforman el proyecto xRAN

De cara a su aplicación en 5G, el grupo 3GPP propone un modelo de red conocido como Cloud-RAN. Este define que las células móviles están formadas por antenas (denominadas Remote Radio Head (RRH) o Remote Radio Unit (RRU)), carentes de inteligencia, que se conectan a las pool de funciones de red mediante un enlace denominado *fronthaul* [5].

El fronthaul se caracteriza por requerir un gran ancho de banda y se puede implementar usando tanto tecnología inalámbrica como cableada. Generalmente, se implementa empleando Fibra Óptica (FO), ya que, es la que proporciona mayor velocidad y menor latencia. La Tabla 2.1 resume las principales ventajas y desventajas del uso de estas dos tecnologías en el fronthaul:

Tabla 2.1: Tecnologías para el fronthaul: ventajas y desventajas.

Tecnología	**Ventajas**	**Desventajas**
Cableada (FO)	Baja latencia y Bandwidth (BW) alto	Caro e inflexible
Inalámbrica	Barato e implementación sencilla	Menor BW y mayor latencia

Estas pool están formadas por un conjunto de elementos definidos por software que se

encargan de realizar las funciones de red de forma virtual, denominados BaseBand Unit (BBU).

Concretamente, 5G categoriza las BBU en dos grandes grupos: uno de ellos destinado a realizar las funciones de procesado de señal (Virtual BBU Resource Pool) y otro para las funciones de control (Control Plane Virtual Resource Pool).

- Virtual BBU Resource Pool ⟶ codificación/decodificación de la señal, aplicación de la Transformada de Fourier, estimación del canal radio, ...
- Control Plane Virtual Resource Pool ⟶ realiza funciones como el control de la conexión, control de portadoras o la asignación de recursos.

La Figura 2.5 sintetiza, de forma gráfica, cómo es la topología propuesta por el grupo 3GPP para la arquitectura *Cloud-RAN* en 5G:

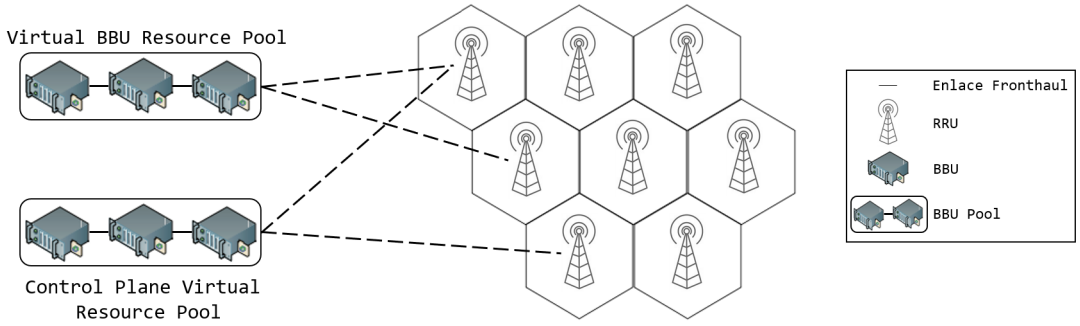

Figura 2.5: Arquitectura Cloud-RAN

A raíz de lo explicado anteriormente, se puede concluir que esta arquitectura permite determinar qué funciones de red se deben virtualizar y cuáles se continuarán realizando en los elementos físicos. Para lograrlo, surge el concepto de *split funcional*, que permite realizar una separación de las funciones entre las RRU y las BBU.

2.3. Software Defined Radio (SDR)

SDR es una tecnología basada en protocolos definidos por software, destinada a las radiocomunicaciones.

A pesar de ser una tecnología que se está desarrollando en los últimos años, este concepto se comenzó a emplear a finales de los años 90. Concretamente, en el año 1993, el ingeniero Joseph Mitola III publicó un artículo [6] en el que argumentaba las razones para emplear soluciones software para el diseño de sistemas radio. Años más tarde, publicó otro artículo donde explicaba el potencial que aportaba SDR para el desarrollo de las redes celulares y WiFi [7].

Respecto a las soluciones definidas por hardware, SDR aporta una mayor flexibilidad e interoperabilidad, ya que, permite actualizar el software sin la necesidad de realizar cambios en la electrónica.

En cuanto a sus potenciales líneas de mercado, Global Industries Analysts destaca su importancia en el ámbito militar, para las comunicaciones espaciales, las comunicaciones V2V e IoT [8].

A lo largo de los siguientes apartados se explica la filosofía y evolución de SDR, así como su arquitectura. Se continuará exponiendo las principales alternativas software y hardware existentes en la actualidad. Por último, se argumentarán las motivaciones que han llevado a decantarse a favor de una de las opciones existentes, srsLTE.

2.3.1. Filosofía y evolución

Uno de los propósitos de SDR es eliminar parte de la electrónica de los dispositivos de radiocomunicaciones, de modo que las funciones que realizaba ese hardware sean efectuadas por software. Esto tiene como consecuencia un abaratamiento de los dispositivos empleados así como una mayor flexibilidad que permite que puedan ser empleados para diferentes protocolos.

Por esa razón, su arquitectura física entremezcla tanto software como hardware, facilitando que determinadas funciones (por ejemplo, la modulación y demodulación de la señal o la selección del canal) no sean responsabilidad de la electrónica [9].

La Figura 2.6 sintetiza los elementos que componen el transceptor, y especifica cuáles de ellos han sido desarrollados gracias al software y cuáles se siguen desplegando físicamente:

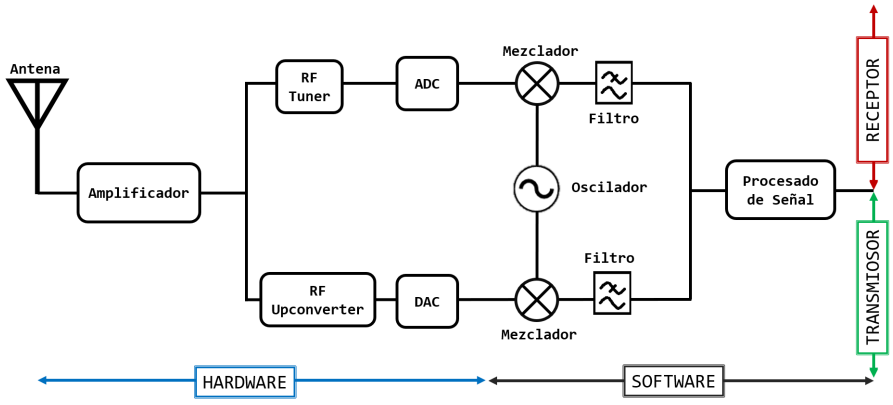

Figura 2.6: Estructura del transceptor: elementos y modos de implementación

La parte superior del esquema se corresponde con el receptor, mientras que la inferior se refiere al transmisor.

Desde el punto de vista de la transmisión, primero se debe procesar la señal que se va a transmitir, es decir, modularla y codificarla. A continuación, la señal banda base se

traslada a la frecuencia intermedia y, mediante el Digital Analog Converter (DAC), se transforma la señal digital en analógica.

Posteriormente, el RF Upconverter transforma la señal analógica en frecuencias del espectro RF (comprendido desde los 3 Hz hasta los 300 GHz) y el amplificador incrementa la señal antes de ser entregada a la antena[1], que se encargará de su envío.

En cuanto a la recepción de la señal, esta se envía a un amplificador de bajo ruido (Low Noise Amplifier (LNA))[2]. Tras ello, el *Tuner RF* transforma las frecuencias RF a la frecuencia intermedia y el Analog Digital Converter (ADC) se encargará de digitalizarla. A posteriori, se traslada la señal a banda base y se filtra. Por último, se realizan las funciones de procesado de la señal (demodulación, decodificación, control y corrección de errores, ...).

Por otro lado, es importante destacar la arquitectura de SDR a nivel software. A esta se la denomina Sofware Communications Architecture (SCA), y se considera un estándar de facto de las telecomunicaciones militares [10].

SCA está compuesta por un conjunto de módulos software procedentes de otras aplicaciones (previamente creadas) y cuyas I/O y dependencias están claramente definidas. Además, es una arquitectura distribuida donde cada uno de sus módulos puede ser ejecutado por componentes diferentes, y su comunicación se garantiza gracias al estándar Common Object Request Broker Architecture (CORBA[3]).

SCA define un protocolo y un entorno para sus módulos. Además, el Core Framework (CF) especifica sus servicios e interfaces, decretando los parámetros requeridos y sus formatos en un fichero Extensible Markup Language (XML).

Los interfaces del CF se pueden clasificar en cuatro grupos:

- Base Application Interfaces: permiten el control y la gestión de los módulos de las aplicaciones implementadas.
- Base Device Interfaces: facilitan el control y la gestión de los dispositivos hardware.
- Framework Control Interfaces (FCI): su implementación se efectúa gracias al software del sistema: permite gestionar y eliminar software de las aplicaciones.
- Framework Service Interfaces (FSI): instalada gracias a módulos software del sistema que aportan funciones para trabajar con ficheros.

La Figura 2.7 representa las capas existentes en esta arquitectura:

[1]Implementada físicamente, debe ser capaz de cubrir un amplio rango de frecuencias permitiendo así que la arquitectura pueda operar en diferentes escenarios (rangos de frecuencias y protocolos de comunicaciones).

[2]Amplificador capaz de incrementar la potencia de la señal recibida sin perjudicar la relación señal-ruido de la misma.

[3]CORBA es un middleware estandarizado por el grupo Object Management Group (OMG) que permite la coordinación de diferentes elementos software que se ejecutan en diferentes hardware.

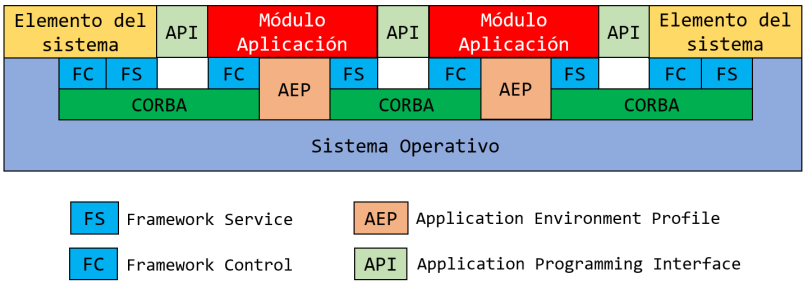

Figura 2.7: Arquitectura SDR

Por último, es importante resaltar los retos a los que se enfrenta SDR [10]:

- **Requisitos computacionales**: Es uno de sus desafíos fundamentales. Es importante conseguir la capacidad computacional adecuada sin comprometer el consumo energético, su tamaño o su peso.

  En el caso de las unidades de SDR que se empleen para desplegar terminales móviles, estos condicionantes son transcendentales: el consumo energético ha de ser bajo, con el propósito de garantizar la duración de las baterías del dispositivo. Asimismo, el tamaño ha de ser pequeño (facilidad de movimiento para el usuario) pero suficiente para evitar sobrecalentamientos del dispositivo.

  Sin embargo, si se usa con el objetivo de desplegar una EB, estos criterios pierden importancia, ya que es un elemento de la red celular que permanece estático. Por lo tanto, se puede conectar a la red eléctrica, consiguiéndose que el consumo energético no sea crítico, así como su peso y tamaño.

- **Plataformas hardware**: Aunque SDR permite desplegar sistemas de comunicaciones radio mediante software, sigue dependiendo de una electrónica que realice determinadas funciones. Para el despliegue del dispositivo físico, existen varias alternativas. Estas posibles soluciones se estudian en el apartado 2.3.3.

- **Seguridad**: SDR es una tecnología flexible que permite alterar su software. Esto es considera un potencial agujero de seguridad, ya que, no se garantiza que solamente será alterado por personal autorizado.

  Además, la instalación de software malicioso puede afectar al sistema operativo del terminal en el que se aloja, o se puede extender al resto de elementos que componen el sistema de comunicaciones establecido. Asimismo, se puede considerar que la instalación de código malintencionado es un potencial *backdoor* para acceder a la información de los usuarios, que reside en el terminal, comprometiéndose así su confidencialidad.

  Para minimizar el riesgo de este problema, se recurre al uso de la criptografía asimétrica, en el que se generan una pareja de claves público-privadas para garantizar la integridad del código y la autenticidad de su creador. Antes de enviar el código a la plataforma, se realiza un *hash* del mismo y se firma empleando la clave privada generada previamente. La Figura 2.8 resume el proceso que se realiza:

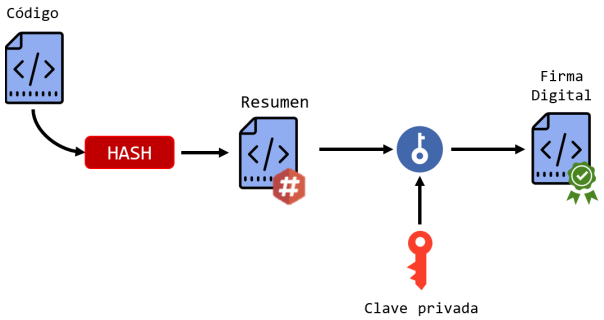

Figura 2.8: Proceso de firma del código SDR

A la plataforma se le envía el código y su firma para que la plataforma pueda validarla y asegurarse que el software no se ha modificado por un tercero.

Una vez recibida la información, la plataforma debe verificar la información recibida. Para ello, se debe comprobar la validez de la firma. Esto se consigue realizando 2 procesos: en primer lugar, realizar el hash del código entregado y, en segundo término, obtener de la firma el resumen del código original (será necesario descifrar la firma empleando la clave pública). Una vez realizados ambos procedimientos, se debe comprobar que ambos resúmenes son idénticos.

En el caso de que difieran, el receptor detecta que el código (o su firma) ha sido modificada por un tercero no autorizado.

La Figura 2.9 resume el proceso a realizar para comprobar la validez de la firma y asegurar que el código no ha sido manipulado:

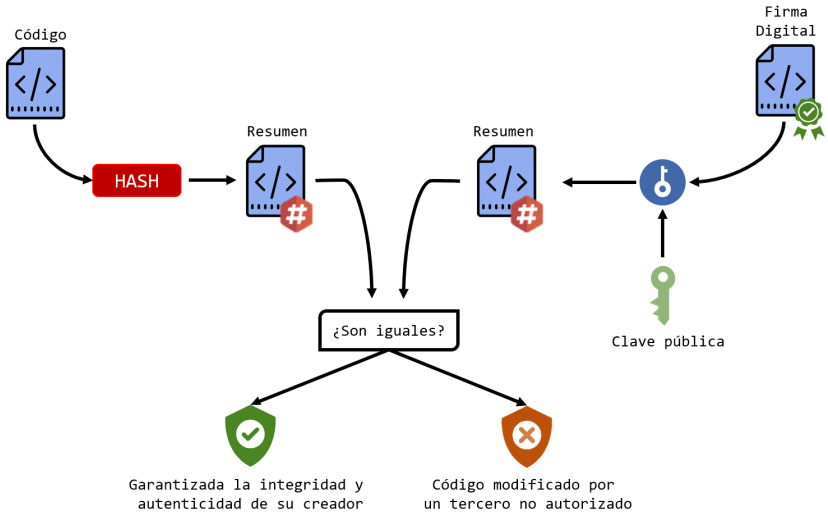

Figura 2.9: Verificación de la firma del código SDR

2.3.2. Alternativas Software

Desde el punto de vista software, en la actualidad existen fundamentalmente 2 potenciales soluciones: *srsLTE* y Open Air Interface (OAI):

- **srsLTE**: Es un proyecto desarrollado por la empresa irlandesa SRS. Se trata de un proyecto de código abierto que permite desplegar una red LTE de Release[4] 8[5].
- **OAI**: Este proyecto, también de código abierto, está siendo desarrollado por el Departamento de Comunicaciones Móviles de EURECOM, una escuela de post-grado y un centro de investigación francés. Mediante esta herramienta, se puede implementar una red LTE de Release 8.6 e incorpora aspectos de la versión 10[6].

Respecto a la primera de ellas, proporciona una librería de código escrita en lenguaje C. Además, recurre frecuentemente al uso de Single Instruction Multiple Data (SIMD), que envía múltiples datos en la misma instrucción. De este modo, se incrementa su rendimiento [11].

Asimismo, proporciona una interfaz que permite utilizar las Universal Software Radio Pherifical (USRP) desarrolladas por Ettus Research[7] aunque puede operar con otras soluciones como Nuand bladeRF.

Proporciona las herramientas suficientes para el despliegue de la red, sus elementos y su configuración. Además, proporciona un sniffer y un analizador, que permite estudiar el rendimiento de las redes desplegadas.

Sin embargo, no es una plataforma finalizada, sino que determinados puntos son inestables y todavía se encuentran en desarrollo.

La otra alternativa, OAI, proporciona una librería de código escrita en C para desplegar una red LTE cuyas características ya se han desarrollado en su totalidad. Además, se trata de un proyecto que en la actualidad se está centrando en el desarrollo de determinadas funciones de 5G, como la comunicación Machine to Machine (M2M) y Cloud-RAN [12].

Al igual que el proyecto srsLTE, puede trabajar con la familia de USRP desarrolladas por Ettus Research o Nuand bladeRF.

Sin embargo, la documentación es escasa y la estructura del código es compleja, lo que dificulta la participación de desarrolladores externos al proyecto. En el caso de srsLTE, su estructuración y documentación ha sido pensada para facilitar el fácil entendimiento del código para aquellos usuarios ajenos al proyecto [11].

[4]El concepto de Release hace referencia a un sistema de numeración de los estándares realizados por el grupo 3GPP. En castellano, conocido como versión.

[5]Esta versión, lanzada en el último trimestre de 2008, fue la primera en estandarizar las redes LTE.

[6]Esta versión estandariza la evolución de LTE: LTE Advanced.

[7]Se trata de una sucursal especializada en SDR de la empresa norteamericana National Instruments.

2.3.3. Alternativas Hardaware

Como ya se ha expuesto anteriormente, SDR precisa de una plataforma hardware que realice determinadas funciones físicas. A esta plataforma se la denomina Commercial Off-The-Shelf (COTS).

Para poder determinar qué tecnologías son más adecuadas utilizar, se describen los principales criterios a tener en cuenta a la hora de tomar la decisión [9]:

- **Flexibilidad & Reconfigurabilidad**: Capacidad de modificar los protocolos y algoritmos del interfaz radio realizando cambios en el software.
- **Capacidad computacional**: Velocidad de procesado de la plataforma. En este contexto, se mide en Giga Operations Per Second (GOPS).
- **Eficiencia energética**: Consumo total de potencia (orden de cientos de mW) de la plataforma, parámetro vital en despliegues IoT y redes celulares.
- **Coste**: Gastos totales incluyendo el estudio de mercado, desarrollo y compra del hardware.
- **Adaptabilidad**: Capacidad de la plataforma de ajustarse a los requisitos de la red y del tráfico que viaja en ella.

De acuerdo a estos parámetros, se describen brevemente 3 tecnologías para el despliegue de la plataforma [9]:

- **General Purpose Processor (GPP)**: Conocido comúnmente como un microprocesador, esta tecnología es utilizada por plataformas como KUAR, Sora o USRP. Su flexibilidad y facilidad de programación la convierten en una de las soluciones más empleadas en el ámbito académico.

  Su rendimiento se ha incrementado notablemente en los últimos años donde se ha logrado aumentar el número de instrucciones por ciclos de reloj, gracias a mejoras tecnológicas y la introducción del paralelismo (donde se combinan varios microprocesadores dando lugar a los *multi-core GPP*).

  Sin embargo, esta tecnología no es adecuada para sistemas con requisitos de tiempo real debido a que no tiene la suficiente capacidad computacional.

  Con el objetivo de minimizar este inconveniente, una potencial solución consiste en añadir un co-procesador. Generalmente, esta función la suple una Graphics Processing Unit (GPU), que son elementos cuya capacidad computacional es mayor que la de los microprocesadores. Este hecho, unido a su alta eficiencia energética, facilita la división de las funciones entre ambas unidades (el GPP se encargará del transporte de la información desde la memoria y la GPU de aplicar los algoritmos de procesado). COTS como Sora o USRP utilizan esta combinación.

- **Digital Signal Processor (DSP)**: A diferencia de las soluciones basadas en GPP, esta tecnología es más rápida y eficiente en la ejecución de determinadas funciones como son: el filtrado, la codificación y la modulación de señales.

  Para el despliegue de sistemas reales, se emplea esta tecnología, ya que sus altos prestaciones computacionales y eficiencia energética la convierten en un recurso óptimo para el despliegue de EB y dispositivos móviles. Un ejemplo de plataforma que utiliza esta tecnología es Atomix.

- **Field-Programmable Gate Array (FPGA)**: Este método consiste en utilizar un hardware programable que se caracteriza por su flexibilidad para adaptarse a diferentes aplicaciones, presenta un tiempo necesario para su reconfiguración bajo (orden de milisegundos) y tiene un alto rendimiento computacional. Asimismo, es una tecnología cuyo consumo energético es bajo y portable. Sin embargo, esta solución es más costosa que las anteriomente descritas. USRP es un ejemplo de una plataforma que emplea esta tecnología.

2.3.4. Elección software & hardware: motivaciones

Aunque se han expuesto en los apartados anteriores de forma independiente, la elección de la plataforma software y del COTS que se va a utilizar está relacionada. Esto se debe a que el software no ofrece soporte para cualquier tipo de electrónica.

Respecto a la plataforma software, se ha decidido que se va a emplear *srsLTE* debido a su mejor organización, y a la extensa documentación del código. Ofrece así un entorno más amigable para poder realizar modificaciones y desarrollos en el mismo.

Al decantarse por este programa, se limita el tipo y número de posibles plataformas hardware que se pueden utilizar. Principalmente, *srsLTE* puede operar con los dispositivos desarrollados por Ettus Research (USRP) o los dispositivos desarrollados por Nuand (bladeRF).

Ambas tecnologías se basan en el uso de FPGAs. Como ya se explicó anteriormente, esta clase de solución se caracteriza por su flexibilidad de adaptación a diferentes aplicaciones, su rapidez de reconfiguración y su alto rendimiento computacional. Además, su consumo energético es bajo y se trata de un dispositivo pequeño (lo que facilita su movilidad). Gracias a estas características, se convierte en una solución óptima para poder desplegar los elementos que componen una red móvil. Su principal hándicap reside en que su coste es mayor que el resto de las alternativas que se expusieron en el Apartado 2.3.3.

Ettus Research ofrece un hardware especializado que se comunica con el Sistema Operativo (SO) del host mediante Ethernet o USB, y ofrece un interfaz para el control de la USRP denominado USRP Hardware Driver (UHD). En concreto, el modelo B210 se trata de una solución que ofrece un elevado rendimiento y, además, es compatible con un gran número de entornos de simulación (Matlab, Simulink y GNU Radio), sistemas operativos (Windows, Linux y MacOS) y softwares SDR (srsLTE, Sora u OpenAirInterface) [13].

Nuand ofrece la serie BladeRF para poder desplegar sistemas de radiocomunicaciones por

software. En concreto, el modelo bladeRF 2.0 micro xA9 THERMAL es aquel que ofrece mejores prestaciones, similares a las del modelo B210 de Ettus, pero no es compatible con el software de srsLTE. El modelo x115 es aquel que aporta las mejores prestaciones y, al mismo tiempo, puede ser utilizado junto a srsLTE. Sin embargo, su rendimiento es muy inferior al del modelo B210.

La Tabla 2.2 resume las principales características del modelo B210 de Ettus Research, que ha sido el que se ha empleado a lo largo del proyecto:

Tabla 2.2: Principales características del modelo B210 [14]

Parámetro	**Valor**
Rango de frecuencias	70 MHz - 6 GHz
Rango BW Analógico	200 kHz - 56 MHz
Resolución ADC/DAC	12 bits
Muestras ADC/DAC	12 Ms/s
Precisión de frecuencia	± 20 ppm
Potencia de salida	>10 dBm
Figura de ruido del receptor	<8 dB
Modelo FPGA	Xilinx Spartan 6 XC6SLX150
Conectividad con host	Puerto USB 3.0
Alimentación	Puerto USB 3.0 y Fuente de Alimentación DC
Otros	Half o Full Dúplex & MIMO 2x2
Usos	Redes celulares, WiFi, TV broadcast, ...

Capítulo 3

Desarrollo del proyecto

A lo largo de este capítulo, se expone el procedimiento que se ha realizado a lo largo del proyecto.

En primer lugar, en la Sección 3.1 se explica el proceso de instalación realizado y se enumeran los principales problemas que se han encontrado a lo largo de la misma.

Posteriormente, en la Sección 3.2 se describe la red desplegada, su proceso de configuración y los inconvenientes encontrados. Ulteriormente, la Sección 3.3 describe los cambios realizados sobre el código *srsLTE* para la consecución de las pruebas.

La Sección 3.4 muestra el proceso de automatización de las pruebas sintéticas. La siguiente sección recoge el proceso realizado para adecuar los ficheros de resultados obtenidos de las pruebas anteriores para poder ser analizados.

Por último, la Sección 3.6 describe el proceso seguido para el despliegue del servidor VLC, así como una explicación de las pruebas que se han realizado.

3.1. Instalación del software srsLTE

Con el objetivo de instalar y poder emplear el software de srsLTE se ha utilizado:

- 2 PC's con SO Ubuntu 16.04 Long Term Support (LTS) y, al menos, un puerto USB 3.0 por ordenador.
- 2 USRP

La versión del SO se justifica en la experiencia. Al momento de realizar la instalación (febrero-marzo de 2020) se intentó en la última versión de Ubuntu: 18.04. Sin embargo, durante el proceso se detectaron problemas de compatibilidad con la librería *libuhd 3.15.0*. Con el objetivo de evitar cambiar la versión del SO se decidió probar con una versión anterior de dicha librería: *libuhd 3.9.0* pero no se logró solucionar dichos problemas.

Por esa razón, se decidió hacer un *downgrade* del SO a la versión 16.04 de Ubuntu (abril de 2016). El uso de esta se justifica en que es la versión LTS inmediatamente anterior a

la 18.04. Al tratarse de una versión de esta clase tiene una mayor estabilidad que el resto y se garantiza su soporte y actualización durante los 5 años posteriores a su salida (en el caso de la 16.04 hasta abril de 2021).

Otro aspecto a destacar de los requisitos enumerados anteriormente es el uso de, al menos, 2 terminales. Esto se debe a que se van a emplear 3 aplicaciones correspondientes a los tres elementos principales de una red LTE: el User Equipment (UE), el ENodeB (ENB) y el Evolved Packet Core (EPC). Un terminal se encargará de ejecutar la aplicación del terminal móvil y otro representará tanto la Estación Base como la parte troncal de la red. El software también permite separar en dos dispositivos diferentes el ENB y el EPC, pero por limitación de equipos no se ha optado por esta opción en el proyecto.

Una vez realizado el proceso de instalación del SO Ubuntu 16.04 en los ordenadores, se ha procedido a instalar el software *srsLTE* siguiendo el proceso detallado en el Anexo I.

A lo largo de la instalación, se han ido encontrando diversos problemas, que se detallan a continuación:

- Las librerías y ejecutables de *srsLTE* se realiza una comprobación de los mismos. Para ello, se realiza un test con el objetivo de comprobar el correcto funcionamiento de un total de 576 ficheros. Durante esta prueba se ha reportado que 8 de ellos han finalizado de forma errónea. A continuación, se muestra la respuesta del comando ejecutado donde se reportan que han fallado 8 pruebas y se especifican cuáles han sido:

```
99% tests passed, 8 failed out of 576
Total Test time (real) = 247.46 sec
The following tests FAILED:
357 - phy_dl_test-p6-t2-q-m27 (Failed)
362 - phy_dl_test-p6-t3-q-m27 (Failed)
367 - phy_dl_test-p6-t4-q-m27 (Failed)
422 - phy_dl_test-p25-t3-m28 (Failed)
427 - phy_dl_test-p25-t4-m28 (Failed)
437 - phy_dl_test-p25-t2-q-m27 (Failed)
442 - phy_dl_test-p25-t3-q-m27 (Failed)
447 - phy_dl_test-p25-t4-q-m27 (Failed)
Errors while running CTest
```

  Este hecho se debe a un defecto del software cuando trabaja con determinadas arquitecturas y los ordenadores con los que se operan tienen una de estas arquitecturas (concretamente, usan una Intel SSE4) [15].

  Sin embargo, este aspecto no es crítico para el funcionamiento de la red que se va a desplegar. El único inconveniente es que habrá un incremento del número de paquetes que se pierdan en la comunicación eNodeB $\rightarrow$ UE, es decir, que el Block Error Rate (BLER) aumentará en Downlink (DL).

- Al igual que ocurrió en el intento de instalación del software en un SO Ubuntu 18.04, se han encontrado problemas de compatibilidad con la versión 3.15 del UHD y se consiguió instalar de forma exitosa la versión 3.9 del mismo.

- Posteriormente, al final del proceso de instalación, ha sido necesario instalar tanto el firmware como los binarios de la FPGA que son compatibles con la versión 3.9 del driver.

Por último, es importante recalcar la necesidad de que los ordenadores dispongan de puertos USB 3.0. Este requisito es totalmente imprescindible, ya que, si no se satisface no se podrá establecer la comunicación entre el terminal y la estación. Concretamente, cuando se inicia uno de los servicios que hacen uso de las USRP (srsUE o srsENB) el software reporta el siguiente problema: *"UHD Warning: The total sum of rates excedees the maximun capacity of the connection. This can cause overflows and underruns"* y no se logra establecer la comunicación entre ellos.

A continuación, se muestra el aviso que reporta el software cuando se conecta la USRP a un puerto no 3.0 y se arranca uno de los servicios:

```
UHD Warning:
The total sum of rates (23.040000 MSps on 1 channels)
exceeds the maxium capacity of the connection. This can
cause overflows (O).
UHD Warning:
The total sum of rates (23.040000 MSps on 1 channels)
exceeds the maxium capacity of the connection. This can
cause underruns (U).
```

Una vez finalizado todo el proceso, ya se tiene listo el software para ser empleado. Una recomendación de los desarrolladores del software es eliminar el CPU Frequency Scaling para mejorar las prestaciones de la red. Para ello, es necesario añadir el siguiente script:

```
$ sudo bash
PATH = /sys/devices/system/cpu/cpu[0-9]*/cpufreq/scaling_governor
for f in $PATH; do
echo performance > $f
done
```

3.2. Red desplegada

Llegado a este punto se dispone de todas las herramientas operativas para poder desplegar la red. En primer lugar, en uno de los ordenadores, se ejecutan las aplicaciones del EPC y del ENB en 2 terminales diferentes.

```
$ sudo srsepc
$ sudo srsenb
```

En el otro PC, se ejecuta el software correspondiente al UE ejecutando el siguiente comando:

```
$ sudo srsue
```

Todas aplicaciones necesitan ser ejecutadas con permisos de superusuario para poder crear *threads* de alta prioridad en el sistema. La Figura 3.1 representa un esquema de todos los dispositivos y aplicaciones necesarios para el despliegue de la red.

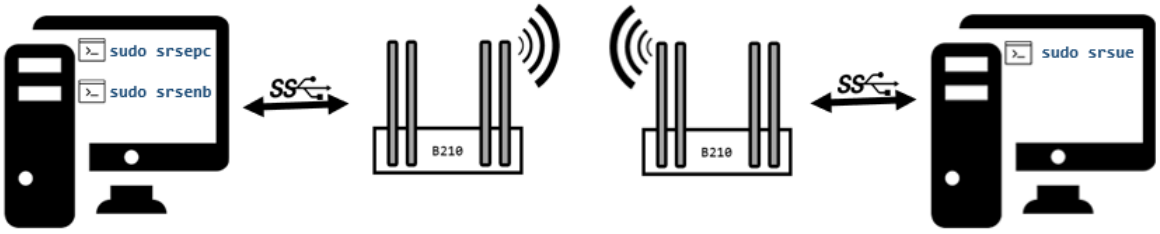

Figura 3.1: Esquema de los dispositivos y sus aplicaciones

Al iniciar el software correspondiente al srsENB, este se conecta a la red troncal (EPC) y comienza a transmitir en *broadcast* para que los terminales móviles se puedan conectar a ella.

Cuando se ejecuta el software del UE se genera un interfaz TUN que es necesario para poder conectarse a la estación y comenzará a buscar a qué estaciones base puede conectarse. En este caso, solamente se podrá conectar a la estación que se desplegó en el otro ordenador.

Una vez encontrada, el terminal procederá con la conexión. Una vez logrado, el usuario y la estación se podrán comunicar entre ellas. A continuación, se muestra un ejemplo de una respuesta que devuelve por pantalla la aplicación cuando el usuario se ha enganchado correctamente a la EB:

```
Attaching UE...
Searching cell in DL EARFCN=1800, f_dl=1865.0 MHz, f_ul=1770.0 MHz
.
Found Cell:  Mode=FDD, PCI=1, PRB=15, Ports=1, CFO=0.2 KHz
Found PLMN:  Id=00101, TAC=7
Random Access Transmission: seq=6, ra-rnti=0x2
Random Access Complete.     c-rnti=0x46, ta=1
RRC Connected
Network attach successful. IP: 172.16.0.2
```

Ejecutando el comando *ifconfig*, se pueden observar la información acerca de los interfaces de red del ordenador. En primer lugar, se muestra la información referente al interfaz de la EB:

```
srs_spgw_sgi Link encap:UNSPEC
HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:172.16.0.1  P-t-P:172.16.0.1  Mask:255.255.255.0
inet6 addr: fe80::879d:3906:318:e5b9/64 Scope:Link
UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:500
RX bytes:0 (0.0 B)  TX bytes:144 (144.0 B)
```

Ahora, se muestra el interfaz del terminal móvil:

```
tun_srsue Link encap:UNSPEC
HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:172.16.0.2  P-t-P:172.16.0.2  Mask:255.255.255.0
```

```
UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:500
RX bytes:0 (0.0 B)  TX bytes:48 (48.0 B)
```

El interfaz `srs_spgw_sgi` se crea cuando se establece la red troncal. Se trata de un Point to Point (PtP) y se precisa que la EB esté conectada al EPC para comenzar el envío de información de red. El interfaz del UE, `tun_srsue`, se genera cuando el usuario consigue conectarse al ENB.

La Tabla 3.1 resume la asignación de direcciones IP que se asignan en la red desplegada:

Tabla 3.1: Resumen del direccionamiento de la red desplegada.

Dirección IP	**Función**
172.16.0.0	Reservada
172.16.0.1	Asignada (por defecto) al eNodeB
172.16.0.255	Dirección broadcast

Por lo tanto, en la red desplegada se disponen de 254 direcciones posibles: una de ellas para la Estación Base (EB) y 253 para los UE. Al terminal se le asigna la dirección IP de menor valor disponible. Por ejemplo, al primer usuario que se conecte se le asigna la dirección IP 172.16.0.2.

Por último, es importante destacar la nomenclatura utilizada. Cuando se habla de enlace descendente (DL) se refiere a la comunicación ENB $\longrightarrow$ UE. Sin embargo, si se hace alusión al ascendente (Uplink (UL)) se trata de la comunicación UE $\longrightarrow$ ENB.

3.2.1. Topología

La Figura 3.2 representa la topología de la red desplegada:

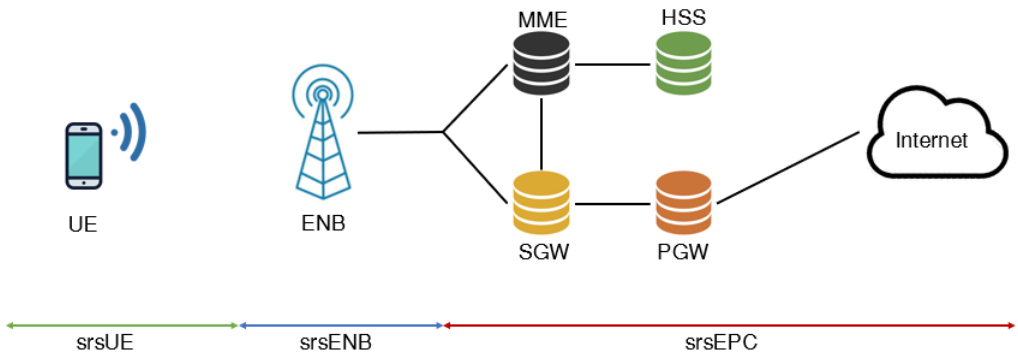

Figura 3.2: Topología de red

Respecto al UE y al ENB, la Figura 3.3 representa la pila de protocolos de cada una de ellas tanto en el plano de datos como en el plano de control:

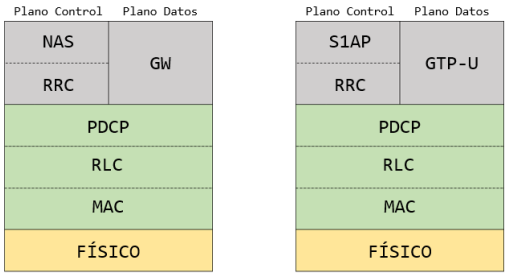

Figura 3.3: Pila de protocolos del UE (izq) y del ENB (dcha)

Como se puede observar, ambos elementos emplean los mismos protocolos tanto a nivel físico (representado en color amarillo) como a nivel de enlace (color verde en la figura) y, en la capa 3 (en gris en la imagen), usan diferentes protocolos. A continuación, se describen brevemente qué funciones realizan estos protocolos:

- **Capa física**: Se encarga del control de potencia y la adaptación del enlace (modulación y codificación empleada). Además, en el UE realiza la búsqueda de EB.
- **Capa de enlace**: Para ambos elementos, se subdivide en tres subcapas.
  - Media Access Control (MAC): Establece prioridades entre los canales lógicos y los multiplexa en Transport Block (TB). Controla la información intercambiada entre el UE y el ENB, realiza las retransmisiones y la corrección de errores utilizando Hybrid Automatic Repeat Request (HARQ).
  - Radio Link Control (RLC): Se encarga de realizar el control de los canales lógicos. Además, concatena, segmenta y reensambla portadoras en Packet Data Unit (PDU).
  - Packet Data Convergence Protocol (PDCP): Garantiza la confidencialidad del tráfico generado (tanto del plano de datos como de control) y la integridad del plano de control. Además, realiza el descarte de duplicados y, opcionalmente, se realiza una compresión de cabeceras, mediante Robust Header Compression (ROHC), de los datos IP.
- **Capa de red**: Respecto a esta, los protocolos empleados son diferentes en el caso del usuario y de la estación base.

  En el caso del UE se utilizan los protocolos RRC y Non-Access Stratum (NAS) en el plano de control, y el Gateway (GW) en el plano de datos. En el EB se usan RRC y S1 Application Protocol (S1-AP) en el plano de control y GPRS Tunnelling Protocol User Plane (GTP-U) en el de datos.
  - RRC: Establece, mantiene y libera las conexiones RRC entre el UE y el ENB. Además, introduce funciones de seguridad para asegurar una comunicación segura entre ambos extremos.
  - NAS: Protocolo encargado de gestionar el tráfico de control entre el UE y el EPC. Empleado para realizar la selección Public Land Mobile Network (PLMN) y el intercambio de información y autenticación del usuario.

- GW: Empleado para crear y mantener los interfaces TUN que permitirán usar al srsUE como una aplicación de usuario que opera con paquetes IP.
- S1-AP: Establece las comunicaciones entre el eNodeB y el EPC relacionadas con la movilidad de los usuarios.
- GTP-U: Permite las comunicaciones de datos entre el eNodeB y el EPC. Encapsula el tráfico IP en paquetes GPRS Tunnelling Protocol (GTP) para ser enviado al Packet Gateway (PGW), que será quien envíe dicho tráfico a Internet.

Por último, es importante destacar la estructura del core de la red implementada, donde se distinguen 4 grandes elementos:

- **Home Subscriber Service (HSS)**: Base de datos que almacena información referente al usuario: identificador, clave, permisos, etc. Es responsable el control de acceso (autenticación + autorización) del usuario a la red.
- **Mobility Managment Entity (MME)**: Encargado de realizar el control de la red. Entre sus responsabilidades, realiza la autenticación de usuarios, asegura la confidencialidad e integridad de la comunicaciones y gestiona la paginación (*paging*).
- **Servicing Gateway (SGW)**: Es el GW que permite la conexión de los usuarios a redes externas. Opera como un router IP y establece sesiones GTP entre el ENB y el PGW.
- **PGW**: Situado en el borde de la red, este elemento permite la comunicación con otras redes. Además, impone los parámetros de QoS de las sesiones GTP. Este se comunica mediante el SGW a través de un túnel.

3.2.2. Configuración

Una vez descrita la red desplegada, el siguiente paso es modificarla y adecuarla para las pruebas que se van a realizar. Para poder manipularla, existen un conjunto de archivos de configuración que permiten modificar los parámetros de los elementos de forma sencilla.

Estos archivos se sitúan en el siguiente directorio:

```
/home/student/.config/srslte
```

La Figura 3.4 resume los archivos existentes en dicho directorio.

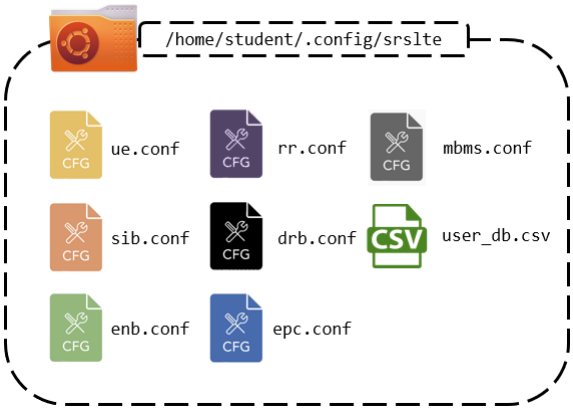

Figura 3.4: Lista de ficheros de configuración de srsLTE

Como se puede observar en la figura anterior, hay un total de 8 documentos: 7 de ellos correspondientes a archivos de configuración y una base de datos (*user_ db.csv*).

Esta base de datos se actualiza cada vez que un nuevo usuario se une a la red. Ahí se almacena información referente a su IMSI, el algoritmo de autenticación utilizado, su clave, su QCI y la estrategia que se empleará para asignarle una dirección IP.

A lo largo de los siguientes puntos se condensan los principales grados de libertad que, mediante la modificación de estos archivos, permiten adaptar la red:

- **Configuración del UE**: Para ello, se emplea el fichero *ue.conf*, que permite principalmente modificar los parámetros RF, de simulación del canal (efecto Doppler, delays,...), la USIM (algoritmo de autenticación, clave, estrategia de direccionamiento deseado, etc) y parámetros físicos. Además, permite configurar los directorios donde se desean que se almacenen las capturas, trazas y logs obtenidos.
- **Configuración del ENB**: Se disponen de 4 documentos para su adecuación. Es importante destacar que estos ficheros no son independientes, por lo que, la realización de cambios en alguno de ellos puede suponer la necesidad de alteraciones de los otros.
  - enb.conf: Preparado para cambiar los parámetros RF y la configuración de los logs, trazas y capturas. También permite configurar el *scheduling* de paquetes en la red, los parámetros para simular el canal, etc. Asimismo, permite configurar la frecuencia de trabajo de la red y determinar el número de Physical Resource Block (PRB) que va a asignar la red a los usuarios.
  - sib.conf: Permite adecuar las características de los System Information Block (SIB).
  - rr.conf: Destinado a la configuración de los recursos radio.
  - drb.conf: Su objetivo es permitir la configuración de los Data Radio Bearers (DRB).
- **Configuración del EPC**: Para lograrlo, se dispone del fichero *epc.conf*, que permite adaptar los parámetros del HSS, el MME, el túnel que comunica el SGW y el PGW.

También, al igual que en los casos de los elementos anteriores, se permite indicar en qué directorios se quieren guardar las trazas, las capturas y los logs obtenidos.

- **Configuración del Multimedia Broadcast Multicast Service (MBMS)**: Archivo centralizado en la configuración de la comunicación broadcast en la red desplegada. Sin embargo, para completar la configuración es necesario modificar los archivos *ue.conf* y *enb.conf*.

Una vez conocidos qué grados de libertad otorgan los ficheros de configuración, el siguiente paso es adecuarlos para la obtención sistemática de resultados.

Para evitar problemas a la hora de localizar características que no funcionen o que tengan algún problema, se ha determinado realizar comprobaciones recurrentes del estado de la red realizando pocos cambios.

En primer lugar, se han modificado las configuraciones relacionadas con los *logs* de los tres elementos de la red. Los *logs* son unos ficheros de elevada utilidad destinados a facilitar el proceso de monitorización de la red. Entre otros aspectos, estos archivos almacenan información acerca de errores que se produzcan en el sistema desplegado.

El formato de estos ficheros es independiente del tipo de elemento al que se refiera. Su estructura se muestra a continuación:

```
[log]
all_level = error
all_hex_limit = 32
filename = /home/student/Desktop/logs/<ue,enb,epc>.log
file_max_size = -1
```

La variable *all_level* permite determinar qué tipo de información se desea que se almacene en este ficheros. Los datos que pueden ser almacenados se corresponden a errores (*"error"*), avisos (*"warning"*), información de los procesos que se efectúan (*"info"*), etc. Asimismo, permite la opción de realizar una monitorización diferente en función de la capa en cuestión (por ejemplo, *"mac_level = error"* permite inspeccionar los fallos que se produzca en el nivel RRC).

En este caso, se va a recoger en los *logs* todos los errores que se produzcan en la red independientemente de la capa en la que tienen lugar.

El parámetro *"filename"* sirve para indicar el nombre que se desea dar al documento y el path donde se desea almacenar. En la red desplegada, se ha almacenado la información en un fichero (cuyo nombre se corresponderá con el elemento que monitoriza) situado en la carpeta *logs* del escritorio.

El siguiente argumento, *"file_max_size"*, permite determinar el tamaño máximo del fichero (en kB). De este modo, si la información recogida supera ese límite, se fragmentará la información en tantos documentos como sea necesario. En este caso, se ha optado por asignarle la opción -1, indicando así que no hay un tamaño máximo de fichero y nunca será necesario fragmentar los datos recogidos.

Posteriormente, se ha modificado la configuración de los ficheros relacionada con las capturas. Como se puede observar a continuación, es necesario habilitarlas e indicar dónde

se quiere que se almacenen:

```
[pcap]
enable = true
filename = /home/student/Desktop/captures/<ue,enb,epc>.pcap
```

Una vez realizados estos cambios, ya se pueden realizar capturas en la red pero para poder analizarlas es necesario efectuar algunos cambios en *Wireshark* (véase el Anexo II).

El siguiente paso a realizar es configurar la información de las trazas. Estas aportarán información útil acerca del enlace radio establecido y se ha empleado como una fuente de información en las pruebas realizadas. En el Anexo III se muestra el formato de estas trazas, así como una breve explicación de cada uno de los parámetros que aporta.

A continuación, se muestra un ejemplo de la configuración de las trazas. En el caso del UE, los campos para su configuración se recogen en la extensión [general] pero, para el ENB, se sitúan en el apartado [expert]:

```
metrics_csv_enable  = true
metrics_period_secs = 1
metrics_csv_filename = /home/student/Desktop/traces/<ue,enb,epc>.csv
```

El campo *"metrics_period_secs"* permite escoger cada cuántos segundos se quiere guardar información acerca de las métricas del enlace radio UE $\longleftrightarrow$ ENB. Aunque su valor puede ser decimal, se ha decidido almacenar información cada segundo.

El siguiente paso a efectuar es adecuar el rango de frecuencias a la que se va a operar en la red. Esta información se recoge en el apartado [rf] de los ficheros de configuración *ue.conf* y *enb.conf*:

```
[rf]
...
dl_earfcn = 1800
...
```

Esto se consigue indicando en qué E-UTRA Absolute Radio Frequency Channel Number (EARFCN)[1] se quiere que trabaje la red. En este caso, se ha asignado el 1800, cuyas principales características se exponen en la Tabla 3.2:

Tabla 3.2: Características EARFCN 1800

Duplexado	FDD
Frecuencia DL (MHz)	1865
Frecuencia UL (MHz)	1770
BW Disponibles (MHz)	1.4,3,5,10,15,20

Asimismo, se pueden asignar frecuencias que no se correspondan con ningún EARFCN.

[1]Número que permite identificar la banda LTE y la frecuencia de las portadoras (UL y DL) a la que opera una determinada red. La correspondencia entre su valor y las frecuencias de trabajo está estandarizado por el 3GPP en la Release 8 (Octubre 2008).

Esto se consigue añadiendo los siguientes campos donde se especifican las frecuencias de trabajo (en Hz) para el enlace descendente y ascendente:

```
[rf]
...
dl_frequency = 400e6
ul_frequency = 450e6
...
```

Sin embargo, se ha optado por ejecutar la primera opción y trabajar con el EARFCN.

Es importante destacar que se ha de configurar el EARFCN que se va a emplear tanto en el móvil como en la estación. Si esto no se realizase, el UE sería incapaz de conectarse a la red desplegada.

Por último, se ha configurado el número de PRB (ver Anexo IV) que la EB ofrece a sus usuarios. Como se ha detallado en la Tabla 3.2, en el EARFCN 1800 se puede trabajar a 1.4, 3, 5, 10, 15 y 20 MHz; que se corresponden a 6, 15, 25, 50, 75 y 100 PRB respectivamente.

Como se ha explicado en el Anexo IV, al incrementar el número de PRB aumenta el ancho de banda que ofrece la red y la velocidad de transmisión de información en la misma. Este parámetro es crítico, ya que, la capacidad de la red para poder ofrecer un determinado número de recursos viene determinada por la capacidad computacional de los PCs que se han empleado.

Para configurar este parámetro, se ha de modificar el campo *n_prb* de la extensión [enb] del fichero de configuración *enb.conf*:

```
[enb]
...
n_prb = 6 # Posibles valores: 6, 15, 25, 50, 75, 100
...
```

Asimismo, es necesario configurar el parámetro *prach_freq_offset*[2] del fichero *sib.conf* (línea 50):

```
43 prach_cnfg =
44 {
45         root_sequence_index = 128;
46         prach_cnfg_info =
47         {
48                 high_speed_flag = false;
49                 prach_config_index = 3;
50                 prach_freq_offset = 2;
51                 zero_correlation_zone_config = 5;
52         };
53 };
```

[2]Esta variable permite determinar la localización del preámbulo del PRACH (Physical Random Access Channel) en el dominio de la frecuencia.

En el que caso de que la red trabaje con 6 PRB, se deberá asignar a esta variable el valor 0 y, en cualquier otro caso, se le dará el 2.

3.2.3. Problemas en la configuración

Los problemas que se han encontrado en la configuración de la red se han producido a la hora de indicar el número de PRB que el eNodeB ofrece. Concretamente, para un valor superior a los 50 PRB se produce un fallo en el radioenlace y no se consigue enganchar el usuario a la red.

Si se observa la información de control que se devuelve por pantalla al ejecutar el comando para iniciar el software del UE se puede comprobar que el usuario es capaz de encontrar la celda a la que se debe unir pero no es capaz de lograrlo:

```
Found Cell:  Mode=FDD, PCI=1, PRB=75, Ports=1, CFO=-0.2 KHz
Found PLMN:  Id=00101, TAC=7
...
Warning: Detected Radio-Link Failure
```

Asimismo, gracias a la información devuelta por los logs, que fueron configurados para almacenar toda clase de errores y problemas que se puedan producir en la red a cualquier nivel, se pudo analizar este problema más en detalle.

A continuación, se explican las dificultades que se han encontrado a lo largo del intento del terminal de unirse a la red desplegada. Para su entendimiento, se mostrará información de los *logs* donde se detallan los errores producidos. Para facilitar la compresión del lector, se indica en qué dispositivo se ha mostrado dicha información mediante el uso de [ue] y [enb].

En primer lugar, el terminal móvil reporta problemas en la recepción del SIB1[3] (línea 2). Posteriormente, se reportan diversos errores en la búsqueda de la celda a la que se debe conectar el usuario (líneas 3-6):

```
1 [ue]
2 [RRC] Proc "SI Acquire" - Timeout while acquiring SIB1
3 [RRC] Proc "PLMN Search" - Failed due to failed cell search sub-procedure
4 [RRC] Proc "PLMN Search" - PLMN Search completed with an error
5 [NAS] Proc "PLMN Search" - Error while searching for PLMNs
```

Tras ello, la EB notifica que no puede transmitir la respuesta RACH (RAR) dentro de la ventana de transmisión (líneas 2-3). Asimismo, el UE indica que no ha recibido la respuesta (línea 6). El usuario retransmite el RACH[4] con el objetivo de que la estación le responda hasta que alcanza el número máximo de retransmisiones (línea 8) y se indica que no se ha logrado establecer la conexión RRC (línea 9).

```
1 [enb]
```

[3]Bloque de información que transporta parámetros importantes como el identificador de la celda, información necesaria para el proceso de *paging*, PLMN,etc.

[4]Permite sincronizar el enlace UL entre el usuario y la estación.

```
2 [MAC] [1167] SCHED: Could not transmit RAR within the window (RA
3 TTI=1151, Window=10, Now=1167)
4
5 [ue]
6 [MAC] [1164] RA response not received within the response window
7 ...
8 [MAC] [1344] RA: Rx: Maximum number of transmissions reached (10)
9 [NAS] Proc "RRC Connect" - Could not establish RRC connection
```

Estos problemas se justifican en que los ordenadores empleados no cuentan con la suficiente capacidad computacional para desplegar una red LTE con más de 50 PRBs.

3.3. Modificaciones del código srsLTE

Para las pruebas que se han realizado sobre la red desplegada se han empleado dos programas externos: IPERF y VLC. El primero se ha usado para las simulaciones con tráfico sintético y la segunda para su estudio en un entorno real.

Por esta razón, es necesario establecer algún método que permita sincronizar la información obtenida de la red en las trazas y la aportada por el software externo (IPERF o VLC).

El procedimiento que se ha empleado es realizar una modificación en la programación de las trazas para que se incluya un sello temporal que se utilizará para sincronizar ambas fuentes de resultados.

El fichero que se ha de modificar se llama *metrics_csv.cc* que se encuentra en la carpeta *src* del directorio *srsue*. Ejecutando el siguiente comando se logra acceder a la carpeta en la que se encuentra:

```
$ cd /home/student/srsLTE/srsue/src/
```

Este archivo es el encargado de almacenar en un documento todos los datos recopilados acerca de los parámetros físicos del terminal móvil (véase la Tabla 5.2, Anexo III) y se ha modificado su programación para que guarde un *timestamp* por cada métrica que almacene.

Una vez finalizada la programación, se deben ejecutar los siguientes comandos con el objetivo de compilar el código modificado, comprobar que no hay ningún tipo de error, regenerar los ejecutables y librerías alterados y copiar esos ficheros y guardarlos en directorios accesibles para el usuario:

```
$ cd /home/student/srsLTE/build
$ cmake ../
$ make
$ sudo make install
```

A continuación, se muestra un ejemplo de una traza obtenida en el terminal móvil. Como se puede observar en la Tabla 3.3, el primer parámetro que se indica es el sello temporal que se ha incluido en la programación:

Tabla 3.3: Ejemplo de una métrica en el UE tras la modificación del código

timestamp	cc	rsrp	pl	cfo	...
09:30:09	0	-51	51	-39	...

3.4. Automatización de las pruebas con tráfico sintético

Las pruebas se han realizado utilizando el software IPERF, que es una herramienta que permite generar flujos de datos, tanto Transport Control Protocol (TCP) como User Datagram Protocol (UDP), con el objetivo de realizar medidas del rendimiento de la red.

En la red desplegada, se ha instalado el software de IPERF en ambos terminales. El PC que ejecuta el software correspondiente al UE actúa como *cliente IPERF* y el ENB como *servidor*. Cabe destacar que en este escenario el cliente actúa como fuente de datos y el servidor como sumidero. Asimismo, el cliente tiene un *buffer* que se irá llenando cuando la velocidad de fuente sea superior a la velocidad de transmisión de datos a través de la red LTE. La Figura 3.5 muestra de forma gráfica la estructura del sistema establecido así como todos los elementos que intervienen en él:

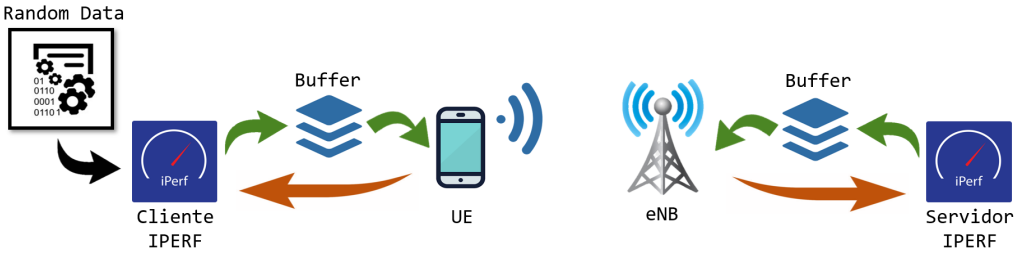

Figura 3.5: Esquema del sistema establecido para las pruebas sintéticas.

En este caso, se ha realizado la evaluación de la red en diferentes escenarios donde se han variado los siguientes parámetros:

- Número de PRB de la EB: 6, 15, 25 y 50.
- Protocolo de capa transporte de los datos enviados: TCP y UDP.
- Velocidad de transmisión de los flujos de paquetes transmitidos.

Concretamente, para cada valor de PRB se ha variado la velocidad con la que transmite IPERF los datos desde el 10 % del throughput máximo con un incremento de un 10 % hasta alcanzar la máxima velocidad de datos teórica. La Tabla 3.4 resume las velocidades iniciales, el paso y la velocidad final, dadas en Mbps, que se han utilizado:

Tabla 3.4: Resumen de velocidades empleadas en la evaluación.

Nº PRB	**Velocidad Inical**	**Incremento**	**Velocidad Final**
6	0.525	0.525	5.25
15	1.125	1.125	11.25
25	1.875	1.875	18.75
50	3.75	3.75	37.5

Asimismo, se ha determinado el tamaño de los datos que se transmitirán en la red. El tamaño de esa información es la equivalente a la que puede enviar, en condiciones ideales, para cada velocidad durante 30 segundos:

$$Datos_{Enviados}(MB) = \frac{Throughput(Mbps) \cdot Tiempo(s)}{8(bits/byte)}$$

La Tabla 3.5 resume el tamaño inicial, el incremento y el tamaño máximo empleado para cada valor de PRB:

Tabla 3.5: Resumen de los tamaños de datos empleados en la evaluación.

Nº PRB	**Tamaño Inicial (MB)**	**Incremento (MB)**	**Tamaño Final (MB)**
6	1.97	1.97	19.7
15	4.22	4.22	42.2
25	7.03	7.03	70.3
50	14.06	14.06	140.6

Para dotar de una mayor robustez y veracidad a la evaluación de la red se ha decidido repetir cada prueba 5 veces. De este modo, en total se han realizado un total de 400 pruebas:

$$Total_{Pruebas} = 5\ repeticiones \cdot 10\ velocidades \cdot 2\ protocolos \cdot 4\ PRB = 400\ repeticiones$$

Para cada uno de los tests realizados es necesario acometer cambios en las configuraciones del ENB y del UE. En el caso de la EB, es necesario modificar el número de PRB y, además, ajustar el valor de la variable *prach_freq_ offset* del fichero *sib.conf*. Respecto al terminal móvil, se debe cambiar el nombre del fichero donde se almacenará la información obtenida del software srsLTE, modificar el nombre del archivo donde se guardará la información reportada por IPERF (su formato se explica en el Anexo V), indicar la cantidad de datos a enviar y a qué velocidad se van a emitir, etc.

Con el objetivo de agilizar el proceso y evitar posibles errores en la configuración, se ha decidido elaborar 2 scripts bash que se encargen de realizar todas estas funciones: UE.sh y ENB.sh.

De este modo, en uno de los terminales se ejecutará el script UE.sh y, en el otro ordenador, se iniciará el software correspondiente al core de la red (srsEPC) y se ejecutará el script que gestiona el ENB (ENB.sh). La Figura 3.6 recoge un esquema que muestra los dispositivos y las aplicaciones que se han ejecutado:

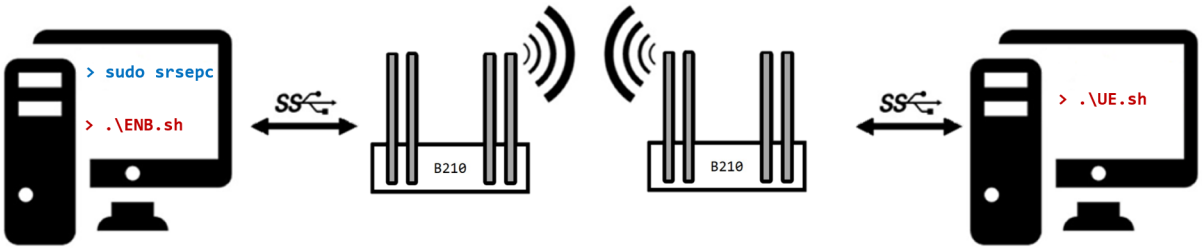

Figura 3.6: Aplicaciones ejecutadas durante las pruebas sintéticas.

A continuación, se explica el procedimiento que realiza cada uno de los scripts elaborados:

UE.sh

En primer lugar, se asigna la velocidad inicial y el primer tamaño de datos que se va a transmitir en función del número de PRB.

Posteriormente, se llama a la función *udp()* que se encargará de realizar las pruebas empleando este protocolo de transporte. Tras ello, se llama a la subrutina *tcp()*. A continuación, se explica qué realizan estas funciones:

- udp(): En primer lugar, genera el nombre del fichero en el que se van a almacenar las métricas obtenidas cuyo formato es el siguiente:

  srsUE-XX-YY-ZZ_ Mbps.csv

  Las XX representan el número de PRB (6, 15, 25, 50), YY el protocolo de nivel de transporte utilizado (en este caso, UDP) y ZZ el throughput al que va a transmitir IPERF la información.

  Posteriormente, se inicia el software del terminal móvil y se comienzan las 5 pruebas que se van a realizar para cada velocidad. Se genera el nombre del fichero donde se va a guardar la información generada por el IPERF, cuyo formato se comenta a continuación. Respecto a la nomenclatura empleada es la misma que la utilizada para el archivo de *srsLTE* pero se añade el término WW que representa el número de prueba en cuestión (1,2,...,5).

  WW-iperf-XX-YY-ZZ-Mbps.csv

  Tras ello, se inicia el cliente IPERF donde se le indica la dirección IP del servidor, es decir, la dirección del ENB (172.16.0.1). Asimismo, se indica la velocidad de fuente, la cantidad de datos que se desean enviar al otro extremo y se indica que se va a emplear UDP en la capa de transporte.

  Una vez finalizada la prueba, se termina el proceso de IPERF y se genera uno nuevo para continuar con la siguiente prueba. Este proceso se repite hasta ejecutar las 5 pruebas programadas.

- tcp(): En términos generales, la filosofía de esta función es la misma que la de la subrutina anterior. Comienza generando el nombre del fichero donde se guardará la información obtenida del terminal móvil con la excepción de que las YY se corresponden al protocolo TCP. Tras ello, genera el nombre del documento donde se guardarán los resultados de IPERF, inicia el software del UE e inicia el cliente IPERF pero indicando que se empleará el protocolo TCP.

Una vez realizado el proceso de las funciones descritas anteriormente descritas, se retorna al programa principal. Este se encargará de incrementar la velocidad y el tamaño de los datos a transmitir y se realizan sus respectivas pruebas. Esto se produce hasta que se realice la evaluación de la red en las condiciones de mayor exigencia, es decir, velocidad y tamaño máximo. Una vez realizadas todas las pruebas, se cierra la aplicación correspondiente al UE.

Ulteriormente, el programa pregunta al usuario si la red ha modificado el número de PRB en la configuración del eNodeB. En caso negativo, se reformulará la pregunta hasta que se produzca el cambio necesario en la red. Cuando se haya producido las modificaciones necesarias, se continuarán las pruebas con el siguiente valor de PRB hasta finalizar la evaluación de la red con 50 PRB.

ENB.sh

En primer lugar, se pide por pantalla al usuario que introduzca el número de PRB que desea asignar a la red. Para evitar que se tenga que reiniciar la red, si el usuario indica que quiere utilizar 50, 75 o 100 PRB se le indica que esos valores no están disponibles y se le vuelve a pedir que introduzca un valor válido. Este proceso se repite de manera indefinida hasta que se introduzca un valor correcto.

Posteriormente, se almacena el valor leído en el fichero *enb.conf* y se ajusta el valor del campo *prach_freq_offset* al número de recursos físicos introducidos.

Por último, se inicia la aplicación correspondiente a la EB y se inicia el servidor IPERF. Ambas aplicaciones estarán ejecutándose un tiempo suficiente para realizar las pruebas y, una vez finalizadas, se terminarán sus procesos.

3.5. Parseo de los ficheros de resultados

El siguiente paso tras obtener los resultados de las pruebas realizadas, es elaborar un entorno que permita coordinar los resultados de ambas herramientas, combinarlos y ajustar su formato para su posterior análisis para la obtención de resultados.

Para lograrlo, se ha desarrollado un programa en Python denominado *parseo.py* que cumple las expectativas requeridas gracias al uso de funciones y herramientas de la librería *Pandas*.

Esta librería provee las herramientas necesarias para poder manipular los datos obtenidos,

eliminando aquellos parámetros que no se consideren útiles y adaptando su formato para que, posteriormente, se puedan analizar y obtener resultados.

parseo.py

A continuación, se muestra en las Figuras 3.7 e 3.8 un esquema que sintetiza la filosofía empleada en el programa:

Figura 3.7: Filosofía de la gestión de resultados TCP.

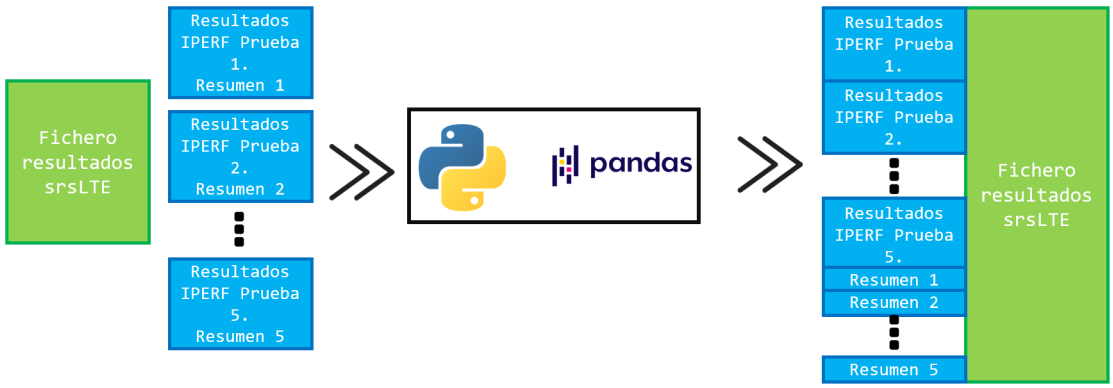

Figura 3.8: Filosofía de la gestión de resultados UDP.

Tal y como reflejan las Figuras 3.7 e 3.8, el programa implementado recibirá como entrada todos los ficheros de resultados de las herramientas *srsLTE* e *IPERF* y se encargará de sincronizar sus resultados para poder combinarlos con el objetivo final de facilitar y adecuarlos para su posterior estudio.

El programa se ha dividido en 5 subrutinas uninivel, donde cada una de ellas realiza una determinada tarea que se especifica a continuación:

- delete_spaces(): Esta función se encarga de preparar cada archivo de resultados IPERF para su posterior evaluación. En primer lugar, debe eliminar los múltiples espacios en blanco entre las variables de los resultados, ya que, si no se eliminan surgirán numerosos problemas con la librería para la lectura de los datos.

Posteriormente, se elimina toda la información adicional que aporta IPERF pero que no tiene ningún tipo de utilidad: "proceso iniciado", "proceso finalizado", "la versión utilizada es...", etc. De este modo, ya se tiene el fichero listo para ser modificado. Asimismo, se devuelve una variable que indica cuántas filas útiles tiene el fichero final, es decir, el número de segundos que ha durado la prueba en cuestión y se almacena en un array.

Además, en el caso de los ficheros cuyo protocolo sea UDP, el resumen que aporta IPERF sobre la prueba se almacena en un fichero externo de forma provisional.

Esta función se ejecuta 5 veces, una por cada repetición realizada de cada prueba.

- iperf_parsing(): Encargada de modificar los ficheros de resultados de IPERF. Se encarga de eliminar columnas cuya información es irrelevante para el análisis: el día y mes devuelto por el comando *ts* para la obtención del timestamp, el identificador (Id.), las unidades de la cantidad de datos enviados y del throughput existente en el enlace. Además, guarda en un array el timestamp para ser empleado en la función srsUE_parsing(). Al igual que la subrutina anterior, se ejecuta 5 veces, una por cada repetición realizada de cada evaluación.

- srsUE_parsing(): Subrutina cuyo propósito es la sincronización de las métricas de *srsLTE* e IPERF. Esto lo logra a partir de los vectores obtenidos en las funciones anteriores: delete_spaces() e iperf_parsing() que se usarán para encontrar las trazas que se corresponden a los resultados obtenidos de IPERF.

- merge(): Se encarga de unir la información del *srsLTE* e *IPERF* en un mismo fichero. Para ello, concatena por columnas los resultados de IPERF y srsTE respectivamente (las Figuras 3.7 y 3.8 ilustran el proceso de forma gráfica). Además, si se trata de un test UDP añade al final del documento la información de los resúmenes de las repeticiones efectuadas.

- fix_finalfile(): Se encarga de eliminar del fichero final la información referente al timestamp estático que se añadió a los ficheros de resultados proporcionados por IPERF.

3.6. Implementación de un servidor VLC

Una vez finalizadas las pruebas con tráfico sintético, se procede a desplegar el escenario para las pruebas con tráfico real.

En concreto, se ha establecido un servidor de streaming de video utilizando VLC al que se conectará un usuario. El servidor se ha instalado en la estación y el terminal móvil actuará como cliente VLC. La Figura 3.9 ilustra el sistema establecido para las pruebas con tráfico real:

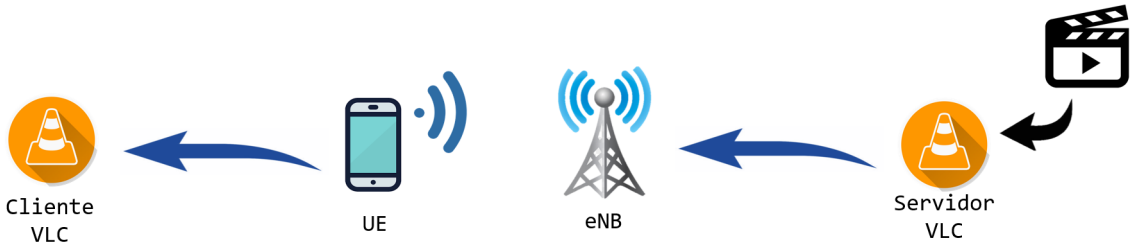

Figura 3.9: Esquema del sistema de streaming de video establecido

Para poder establecer el servidor VLC es necesario ejecutar el procedimiento que se detalla a continuación:

En primer lugar, se ha de abrir la pestaña "Emitir" que se encuentra en la opción "'Medio" del menú principal de la aplicación VLC. Por defecto, se abre una ventana emergente donde aparecen varias pestañas. Concretamente, en "Archivo" se permite elegir qué video se desea emitir al hacer click en el botón "Añadir".

Una vez escogido la fuente de transmisión, se avanza pulsando la opción de "Emitir". Tras esto, se abre una nueva ventana donde no se realizará ningún cambio y se pulsará la opción para avanzar.

Llegado a este punto, se deberá indicar el nuevo destino donde se indicará que es HTTP y se marcará la opción de mostrar en local. La Figura 3.10 muestra la ventana con los cambios necesarios realizados:

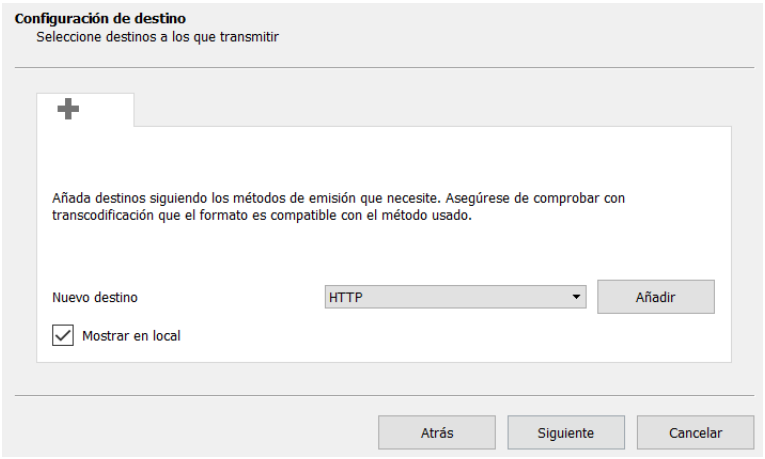

Figura 3.10: Configuración del destino para el streaming

Una vez introducidos los cambios mostrados será necesario hacer click en el botón "Añadir". Al efectuarlo, se avanzará de ventana y se solicitará indicar en qué puerto se desea emitir. En este caso, se ha empleado el mismo que aparece por defecto: el 8080.

El siguiente paso consiste en indicar el perfil que se va a utilizar en la transmisión. En este caso, la opción empleada es: Video - H.264 + MP3 (MP4). Una vez realizado el cambio se continúa avanzando hasta llegar al final sin realizar ninguna modificación.

Una vez establecido el servidor VLC, el procedimiento para que el cliente se conecte es sencillo. En la pantalla principal de la aplicación, se debe abrir la pestaña "Abrir ubicación de red" que se encuentra en la opción media del menú principal.

Se abre una ventana emergente en la que se pide que se introduzca la URL del servidor al que se quiere conectar. La dirección que se debe indicar sigue el siguiente formato: "http://<IP_Servidor>:<Puerto>". En este caso, la dirección empleada es la siguiente: "http://172.16.0.1:8080" siendo la dirección IP la correspondiente al ENB.

Para la consecución de las pruebas, se ha emitido un video [17] con diferentes calidades. En concreto, se ha trabajado con tres calidades de video: 360p, 720p y 1080 y se ha transmitido cada una de ellas en las diferentes configuraciones de PRB posibles en la red (6, 15, 25 y 50). La Tabla 3.6 expone los requisitoss necesarios para poder efectuar las transmisiones de forma correcta.

Calidad	**Resolución**	**Throughput mínimo**
360	640x360	864 kbps
720	1280x720	3.5 Mbps
1080	1920x1080	7.7 Mbps

Tabla 3.6: Relación resolución-throughput en función de la calidad de la emisión

Para el cálculo del throughput mínimo necesario es preciso tener en cuenta que se ha empleado la codificación H.264. Esto implica que el Quality Factor (QF)[5] utilizado es 0.125, es decir, un bit por cada 8 píxeles. También es necesario conocer el número de FPS[6] que se envía en la transmisión: en este caso, para las tres calidades se envían 30 FPS.

A partir de estos datos, se pueden calcular el throughput de la Tabla 3.6 empleando la siguiente fórmula:

$$Throughput_{min.}(bps) = FPS \cdot Resolución \cdot QF$$

[5]Expresa la relación entre el número de píxeles y el número de bits. También conocido como bpp (bits per pixel).

[6]Expresa la frecuencia de envío de fotogramas enviados por unidad de tiempo. En inglés, Frames Per Second.

Capítulo 4

Pruebas y resultados

A lo largo de este capítulo, se exponen los principales resultados obtenidos a lo largo de los experimentos realizados sobre la red desplegada.

La Sección 4.1 recoge los resultados obtenidos de las pruebas realizadas con tráfico sintético. Concretamente, se describen los escenarios en los que se han realizado los expermientos y se detallan los resultados obtenidos.

Ulteriormente, la Sección 4.2. condensa los resultados logrados de las pruebas realizadas empleando el servidor de streaming de video. Precisamente, se detallan los escenarios en los que se ha operado para estudiar la red desplegada y se muestran los resultados conseguidos de las pruebas realizadas.

4.1. Pruebas con tráfico sintético: IPERF

Como ya se ha comentado previamente, estas pruebas se han realizado en 4 escenarios. Cada uno de ellos se corresponden a un despliegue diferente de la red LTE donde se ha variado el número de PRB que asigna el ENB: 6, 15, 25 y 50 PRBs.

Para cada una de estas situaciones, se ha trabajado con dos protocolos de la capa de transporte (TCP y UDP) y, en cada uno de ellos, se ha incrementado progresivamente la velocidad de los datos que emite la herramienta IPERF de acuerdo a los valores teóricos de capacidad máxima de cada configuración.

A continuación, se muestran los resultados obtenidos. Como estos son similares en las diferentes configuraciones se ha decidido mostrar únicamente uno de los escenarios para evitar la repetición en las explicaciones dadas. En concreto, las gráficas mostradas se corresponden a la experimentación realizada para la red con 25 PRB.

La Figuras 4.1 y 4.2 representan la evolución de la velocidad de transmisión real que proporciona la red en función de la velocidad de fuente empleando los protocolos TCP y UDP respectivamente:

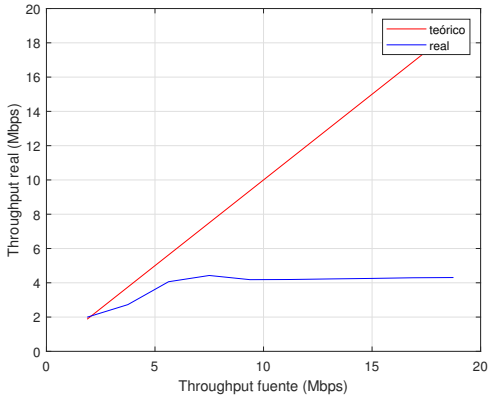

Figura 4.1: Throughput fuente vs red (Protocolo TCP)

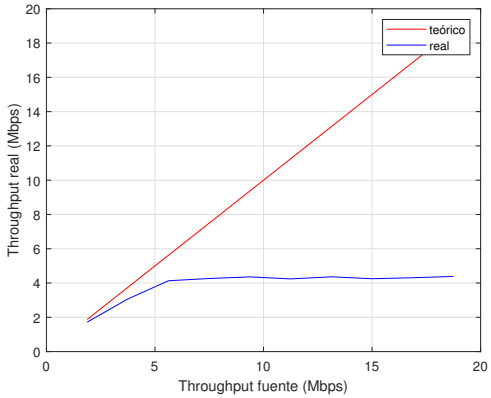

Figura 4.2: Throughput fuente vs red (Protocolo UDP)

En estas figuras, el eje X representa el throughput de fuente de IPERF, medido en Mbps, y el eje Y la velocidad real a la que transmite la red, también medida en Mbps. Cada gráfica contiene 2 curvas: la evolución teórica (representada en rojo) y la evolución experimental (representada en azul).

Desde el punto de vista teórico e ideal, la red debería ser capaz de permitir transmisiones a la misma velocidad que la teórica. Sin embargo, esto es utópico debido a las limitaciones computacionales existentes.

Teniendo esto en cuenta, el comportamiento obtenido experimentalmente se ajusta al esperado: donde, para bajas velocidades, la red es capaz de enviar la información a una velocidad similar a la velocidad de fuente. Sin embargo, a medida que crece la velocidad de fuente, la red alcanza su máxima tasa de transmisión y se mantiene a la misma independientemente del incremento de la tasa de fuente de IPERF.

La Tabla 4.1 muestra las máximas velocidades de transmisión que se alcanza para cada una de las configuraciones:

Tabla 4.1: Relación calidad-throughput en función de la calidad de la emisión

PRB	6	15	25	50
$\mathbf{V}_{max.}$ **(Mbps)**	0.4	2.16	4.3	10

Cabe destacar que las tasas máximas se alcanzan empleando el protocolo UDP, ya que, no es necesario realizar ningún tipo de retransmisiones.

Las Figuras 4.3 y 4.4 representan la variación del número de retransmisiones medias necesarias en TCP y el porcentaje de paquetes perdidos en UDP en función de la velocidad de fuente respectivamente.

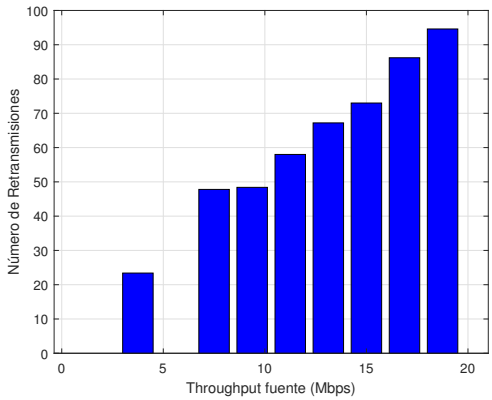

Figura 4.3: Throughput fuente vs Retransmisiones medias (Protocolo TCP)

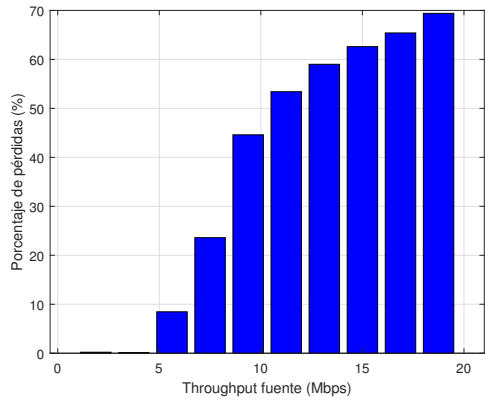

Figura 4.4: Throughput fuente vs % Paquetes perdidos medios (Protocolo UDP)

En la Figura 4.3 se representa en el eje de abcisas la velocidad de fuente en Mbps y en el eje de ordenadas el número medio de retransmisiones necesarias.

Desde un punto de vista intuitivo el resultado esperado sería un crecimiento lineal del número de retransmisiones que se necesitan a medida que aumenta la velocidad de la fuente. Los resultados experimentales se ajustan a los esperados donde existe una correlación directa entre la velocidad de fuente y el número de retransmisiones necesarias.

En la gráfica de la Figura 4.4 se representa en el eje horizontal la velocidad de la fuente y en el vertical el porcentaje de paquetes perdidos en la transmisión UDP.

La tendencia de los resultados es la misma que en el caso de las retransmisiones: una correlación lineal entre el porcentaje de paquetes perdidos y la velocidad de envío de información de IPERF.

A continuación, se representa en las Figuras 4.5 y 4.6 la evolución del buffer, SNR y la velocidad de tranmisión que permite la red con el tiempo para una velocidad de fuente constante empleando los protocolos TCP y UDP respectivamente.

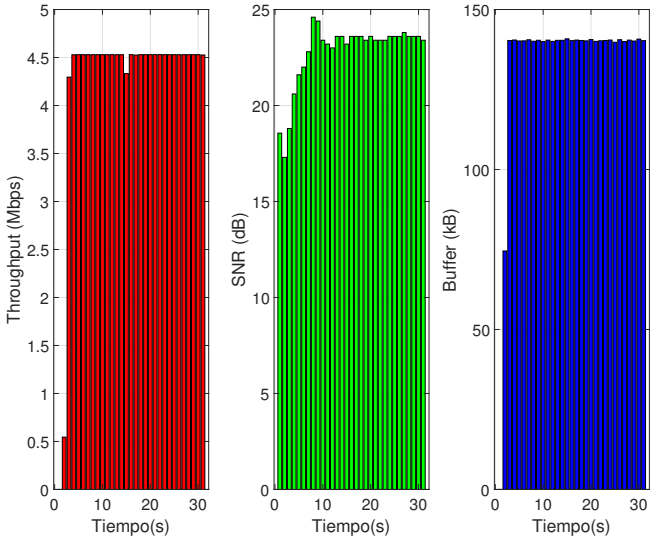

Figura 4.5: Tiempo vs Throughput, SNR y Buffer (Protocolo TCP)

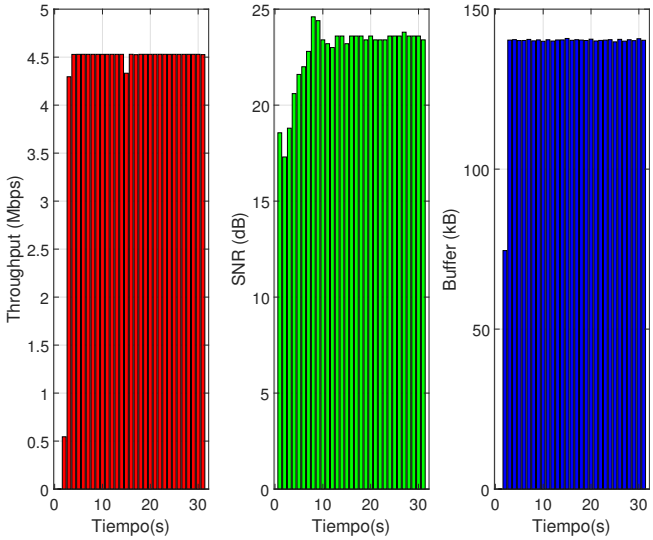

Figura 4.6: Tiempo vs Throughput, SNR y Buffer (Protocolo UDP)

En cada una de las figuras, se muestran tres gráficas. El eje X es el mismo para todas ellas y representa el tiempo medido en segundos. Sin embargo, el eje Y varía en función de la gráfica: en la primera representa el throughpout medido en Mbps, en la segunda el

SNR del enlace radio medido en dB y, por último, en la tercera se representa la cantidad de datos, medidos en kB, existentes en el buffer de transmisión.

En ambos casos, los resultados obtenidos de forma experimental se ajustan a los esperados. Como se puede observar, los tres factores que se representan en las gráficas tienen un comportamiento constante. Asimismo, es importante destacar la relación existente entre ellos.

Una degradación en la relación señal-ruido en el enlace provocaría un descenso del throughput y un aumento de la cantidad de información que se almacena en el buffer. Es decir, que existe una relación entre el SNR y la cantidad de datos que se almacenan en el buffer y hay una relación entre el SNR y la velocidad de transmisión de los datos.

Por último, es importante destacar que se ha detectado una nueva incidencia durante la realización de estas pruebas. Este problema consiste en que la red, con una configuración de 50 PRBs, es incapaz de transmitir la información que se envía cuando la velocidad de la fuente es superior a 26.25 Mbps. Esto se debe a que el buffer genera un cuello de botella en la red y acaba cayendo debido a que se detecta un problema en el radioenlace (*Detected Radio-Link Failure*).

4.2. Pruebas con tráfico real: servidor VLC

Al igual que la experimentación sintética, estas pruebas se han realizado en 4 escenarios, cada uno de ellos correspondiente a las 4 posibles configuraciones de PRB disponibles en la red desplegada.

Para cada una de los diferentes marcos de experimentación, se ha establecido el servidor VLC donde se ha emitido un video con diferentes calidades[1]: 360p, 720p y 1080.

Dada la similitud de los resultados en las diferentes configuraciones, a continuación se muestran los resultados obtenidos para una de las configuraciones con el objetivo de evitar una excesiva repetitividad en las argumentaciones dadas. En concreto, se muestran los resultados para la configuración de 15 PRB.

Las Figuras 4.7, 4.8 y 4.9 recogen, cada una de ellas, dos gráficas donde se representa la evolución del throughput (en Mbps) y del SNR (dB) en función del tiempo (segundos) para cada una de las calidades que se han transmitido.

El eje de abcisas es el mismo para ambas gráficas y se corresponde al tiempo, medido en segundos, mientras que el eje Y se refiere a la velocidad de transmisión de la red (gráfica izquierda) y al ratio señal-ruido del radioenlace (gráfica derecha).

[1] https://test-videos.co.uk/bigbuckbunny/mp4-h264

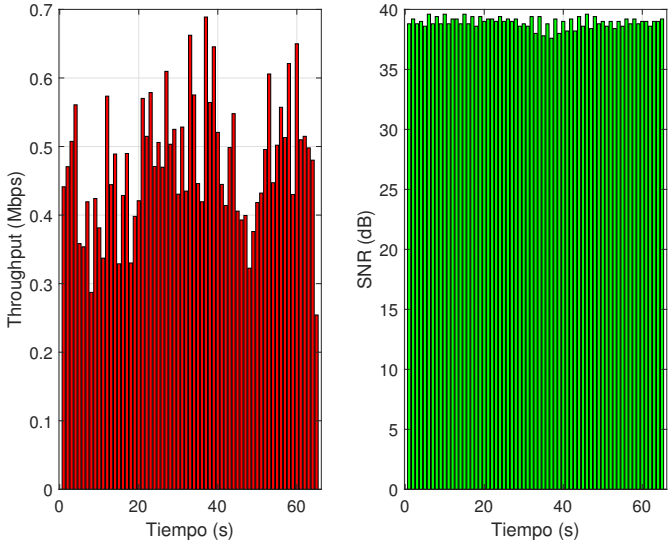

Figura 4.7: Tiempo vs Throughput y SNR (Calidad 360p)

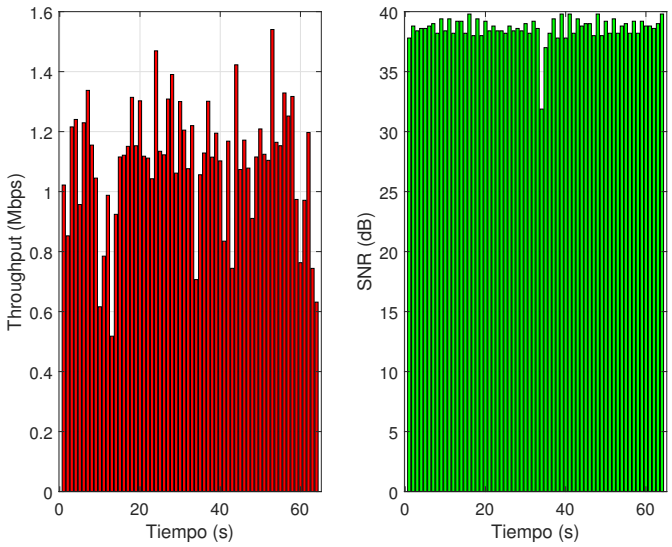

Figura 4.8: Tiempo vs Throughput y SNR (Calidad 720p)

Respecto a la Figura 4.7, se puede observar que la relación señal-ruido permanece constante a lo largo de la transmisión. Sin embargo, la velocidad de la red sufre fluctuaciones constantes.

En cuanto a la Figura 4.8, el comportamiento del SNR es similar al producido en la emisión de video a 360p aunque se produce caídas puntuales del SNR. Al igual que en el caso anterior, el throughput no es constante y se producen fluctuaciones.

Desde el punto de vista visual, ambas retransmisiones cumplen las expectativas. Se caracterizan por tener tiempos de *prebuffering* inferiores a los 10 segundos y la emisión es fluida.

La Figura 4.9 se corresponde a la emisión del video en calidad 1080. Al igual que ocurre

en los casos anteriores, el SNR permanece constante a lo largo del tiempo pero la tasa binaria sufre fluctuaciones.

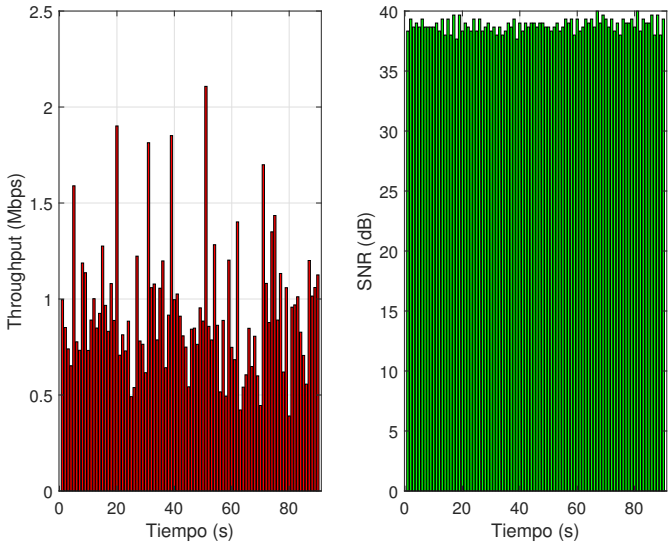

Figura 4.9: Tiempo vs Throughput y SNR (Calidad 1080)

Además, es destacable la calidad de la retransmisión desde el punto de vista visual, donde el tiempo de prebuffering es superior (≈30 segundos). Asimismo, el video no se puede visualizar de forma fluida y se pixela recurrentemente la imagen.

Las fluctuaciones existentes se deben a la inestabilidad a la red, que es causada por las limitaciones computacionales de los recursos empleados para poder desplegar y mantener de forma simultánea la red LTE y el servidor de streaming establecido.

Al igual que en la pruebas sintéticas, a lo largo de las experimentaciones con tráfico real se han detectado varios problemas. Estas limitaciones son consecuencia de la inestabilidad de la red debido a los componentes empleados. Las consecuencias de estos inconvenientes es la imposibilidad de establecer el servidor VLC cuando se emite el video de calidad 1080. En el caso de la configuración de red de 50 Physical Resource Block (PRB)s, este problema se acentúa más, ya que no se puede establecer dicho servidor para ninguna de las calidades de los videos con los que se ha trabajado.

La Tabla 4.2 resume qué pruebas han funcionado y cuáles no:

Tabla 4.2: Resumen de las pruebas exitosas y fallidas.

PRB	**Calidad**		
	360p	720p	1080
6	✓	✓	✓
15	✓	✓	✓
25	✓	✓	✗
50	✗	✗	✗

Capítulo 5

Conclusiones

Las soluciones basadas en SDR se han convertido en una alternativa para el despliegue de redes de comunicaciones en entornos de simulación. Se caracterizan porque la mayoría del despliegue se realiza en software, reduciéndose la dependencia existente de la electrónica.

Además, esta tecnología tiene su repercusión en el desarrollo de 5G donde uno de sus objetivos consiste en trasladar determinadas funciones que se realizaban en la red de acceso a la red troncal y que se efectúen de forma virtualizada.

A lo largo de este trabajo, se ha desplegado una red LTE definida por software empleando la implementación *srsLTE* y se ha estudiado su viabilidad, rendimiento y fiabilidad ante diferentes escenarios.

Durante la configuración, se han localizado diversos problemas. En primer lugar, un incremento del BLER como consecuencia de emplear ordenadores con una arquitectura Intel SSE4. Además, la imposibilidad de desplegar la red con una configuración de 75 y 100 PRB como consecuencia de la limitación computacional de los terminales empleados.

Respecto a los experimentos realizados con tráfico sintético, es importante destacar la estabilidad de red para permitir un *throughput* constante en el tiempo. Sin embargo, para las configuraciones estudiadas se alcanzan velocidades de transmisión de datos muy inferiores a las que se deberían esperar teóricamente. Asimismo, la imposibilidad de trabajar con velocidades de fuente superiores a los 26.25 Mbps en la configuración de 50 PRBs. Ambos problemas son ocasionados debido a las limitaciones computacionales de los equipos empleados.

Por otro lado, se ha evaluado la red empleando tráfico real. Para ello, se ha desplegado un servidor VLC y se ha emitido diferentes *streams* con diferentes calidades para cada una de las configuraciones.

En este escenario, es importante destacar que, a diferencia de las evaluaciones sintéticas, la red pierde estabilidad y es incapaz de permitir un *throughput* constante a lo largo del tiempo. Esto se debe a que, en esta situación, los elementos computacionales involucrados tienen un mayor trabajo que en las otras pruebas. Estos deben desplegar y mantener la red LTE, desplegar el servidor y cliente VLC. Además, el ENB debe codificar el *streaming* y enviárselo al cliente, que se encargará de su decodificación y mostrárselo al usuario final.

Esta sobrecarga alcanza su punto álgido cuando se trata de emitir en la configuración de 50 PRB donde los dispositivos no tienen la suficiente capacidad para cumplir todas las tareas que ha de realizar.

Con el objetivo de resolver estos problemas y reducir la sobrecarga, se propone utilizar 3 terminales: uno para el UE, otro para la EB y un tercero para desplegar los servicios. De este modo se lograría distribuir todas las tareas que se han de realizar consiguiéndose así un comportamiento más previsible de la red desplegada.

5.1. Líneas futuras

Como se ha comentado previamente, en este proyecto se ha evaluado el comportamiento de una red LTE desplegada mediante SDR. De este modo, surgen una serie de líneas de trabajo, donde destacan:

- Despliegue de la red empleando 3 terminales: como ya se ha explicado en la sección anterior, se propone desplegar la red utilizando 3 ordenadores. De este modo, se logra distribuir el trabajo que ha de cumplir cada uno de ellos y se reducen los problemas causados por las limitaciones de los mismos.
- Análisis más profundo de los requisitos computacionales en función de cada configuración.
- Modificación de la arquitectura LTE.
- Despliegue de la red LTE empleando dispositivos reales para los UE, es decir, *smartphones*. De este modo, se logra dotar de una mayor veracidad al entorno de emulación desplegado.

Referencias

[1] R. Sood and A. Garg, “Digital Society from 1G to 5G: A Comparative Study”, M.M. Institute of Computer Technology & Business Management, Maharishi Markandeshwar University.

[2] Ramón Agüero, “Tema 4 - Dimensionado de sistemas celulares”, Redes de Comunicaciones, UC. “https://www.tlmat.unican.es/siteadmin/submaterials/3240.pdf” (visitado el 30 de mayo de 2020)

[3] Luis Muñoz, “Tema 2 - El Sistema GSM”, Redes Inalámbricas, Universidad de Cantabria. “https://www.tlmat.unican.es/siteadmin/submaterials/3193.pdf” (visitado el 30 de mayo de 2020)

[4] “An Introduction to Network Slicing”, GSMA 2017.

[5] S. Abdelwahab, B. Hamdaoui, M. Guizani and T. Znati, “Network function virtualization in 5G”, in IEEE Communications Magazine, vol. 54, no. 4, pp. 84-91, April 2016.

[6] J. M. Mitola, “Software Radios Survey, Critical Evaluation and Future Directions”, IEEE Aerospace and Electronic Systems Magazine, vol. 8, no. 4, pp. 25–36, apr 1993.

[7] W. H. W. Tuttlebee,“Software-defined radio: facets of a developing technology”, IEEE Personal Communications, vol. 6, no. 2, pp. 38–44, apr 1999.

[8] Global Industry Analysts Inc, “Software Defined Radio (SDR)- Global Strategic Business Report”, Tech. Rep., 2017.

[9] R. Akeela, B. Dezfouli, “Software-defined Radios: Architecture, state-of-the-art, and challenges”, Computer Communications, Volume 128, 2018, pp. 106-125.

[10] T. Ulversoy,“Software Defined Radio: Challenges and Opportunities”, in IEEE Communications Surveys & Tutorials, vol. 12, no. 4, pp. 531-550, Fourth Quarter 2010.

[11] I. Gomez-Miguelez, A. Garcia-Saavedra,P. Sutton, P. Serrano, C. Cano and D. Leith, “srsLTE: an open-source platform for LTE evolution and experimentation”, 2016, pp. 25-32.

[12] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet. 2014. “OpenAirInterface: A Flexible Platform for 5G Research”. SIGCOMM Comput. Commun. Rev. 44, 5 (October 2014), pp. 33–38.

[13] A. Gaitán, “Desarrollo de un sistema para simulación, generación y recepción de señales 4G y 5G”. Universidad Politécnica de Madrid. pp. 23-24.

[14] Ettus Research, B210 Model Features & Datasheet: "https://www.ettus.com/all-products/UB210-KIT/"

[15] https://github.com/srsLTE/srsLTE/issues/423 (visitado el 17/05/2020)

[16] J.M.Hernando, “Tema 6: Capa Física LTE”, pp.44, Cátedra Isdefe, UPM. (visitado el 29 de junio de 2020)

[17] Big Buck Bunny, url: https://peach.blender.org/

ANEXO I: HOW TO INSTALL

En primer lugar es necesario descargar el paquete de software SRS del repositorio *srsLTE*:

```
$ sudo add-apt-repository ppa:srslte/releases
```

Tras ello, se ejecuta el siguiente comando que actualiza los paquetes del SO:

```
$ sudo apt-get update
```

Se instala el paquete software de srsLTE descargado previamente:

```
$ sudo apt-get install srslte -y
```

Posteriormente, se instalan las siguientes librerías:

```
$ sudo apt-get install cmake libfftw3-dev libmbedtls-dev \
libboost-program-options-dev libconfig++-dev libsctp-dev
```

Se descarga el repositorio srsLTE de GitHub. En el caso de que falle este comando, se debe revisar que se tenga instalado el paquete *git* en el ordenador:

```
$ git clone https://github.com/srsLTE/srsLTE.git
```

Se cambia al directorio *srsLTE* que se ha generado en la instalación:

```
$ cd srsLTE
```

Se crea el directorio *build* y se accede a él:

```
$ mkdir build
$ cd build
```

Mediante el siguiente comando se genera el sistema de compilación (Buildsystem):

```
$ cmake ../
```

Genera todos los ejecutables y librerías del proyecto:

```
$ make
```

Una vez generados las librerías y los ejecutables, se realiza una comprobación del correcto funcionamiento de 576 ficheros pero, como se ha explicado en la sección 3.1, fallan 8 de las pruebas:

```
$ make test
```

La siguiente instrucción se usa para copiar los ficheros generados anteriormente y almacenarlos en directorios accesibles:

```
$ sudo make install
```

Se instalan los archivos de configuración mediante los siguientes comandos:

```
$ sudo srslte_install_configs.sh user
$ sudo srslte_install_configs.sh service
```

A continuación, se ejecutan los servicios instalados:

```
$ sudo srsepc
$ sudo srsenb
$ sudo srsue
```

Una vez probado el funcionamiento de los servicios, es necesario descargar e instalar una serie de paquetes y librerías que se necesitan para garantizar el correcto funcionamiento de la red desplegada:

```
$ sudo apt-get -y install git swig cmake doxygen build-essential \
libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev \
libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc \
libcppunit-1.13-0v5 libcppunit-dev libcppunit-doc ncurses-bin \
cpufrequtils python-numpy python-numpy-doc python-numpy-dbg \
python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc \
libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev \
1python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev \
libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg \
libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake \
autoconf libtool python-dev libfftw3-dev libcppunit-dev \
libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev \
python-wxgtk3.0 git-core libqt4-dev python-numpy ccache \
python-opengl libgsl-dev python-cheetah python-mako python-lxml \
doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev \
libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core \
wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 \
liborc-0.4-dev libasound2-dev python-gtk2 libzmq-dev libzmq1 \
python-requests python-sphinx libcomedi-dev python-zmq \
python-setuptools
```

Es necesario descargar la versión LTS 3.9 del UHD de su repositorio:

```
$ sudo add-apt-repository ppa:ettusresearch/uhd-3.9.lts
```

Posteriormente, se instalan las librerías que se han obtenido del comando anterior:

```
$ sudo apt install libuhd-dev
$ sudo apt install libuhd003
```

Se reinicia el ordenador para que se completen las instalaciones de las librerías:

```
$ sudo reboot
```

Se instalan los servicios de UHD:

```
$ sudo apt install uhd-host
```

Mediante el siguiente comando se crean los vínculos y caché necesarios a las librerías compartidas que se encuentren en los directorios /lib y /usr/lib. Es decir, todas las librerías que se han instalado a lo largo del proceso de instalación descrito en este anexo:

```
$ sudo ldconfig
```

Por último, es necesario instalar el firmware y los binarios de la FPGA compatibles con la versión del UHD instalados en el ordenador previamente:

```
$ sudo /usr/lib/uhd/utils/uhd_images_downloader.py
```

ANEXO II: WIRESHARK

srsLTE permite realizar capturar el tráfico en la capa MAC y en la capa NAS. En la primera capa, se capturará tanto el tráfico de datos como de control y, si se ha configurado para ello, este tráfico estará cifrado. Sin embargo, en la capa NAS solamente se captura tráfico de datos y siempre aparecerá en claro.

Para poder analizar el tráfico capturado, srsLTE permite visualizarlo en Wireshark. Para poder verlo de forma correcta y observar todos los campos de los niveles es necesario configurar Wireshark.

Su configuración consiste en añadir una serie de entradas donde se relaciona el protocolo con un código numérico que utiliza Wireshark.

Para llegar hasta esa tabla se debe ir a *Edit* $\longrightarrow$ *Preferences*. Al hacer click en esta pestaña se abrirá una ventana semejante a la mostrada en la Figura 5.1:

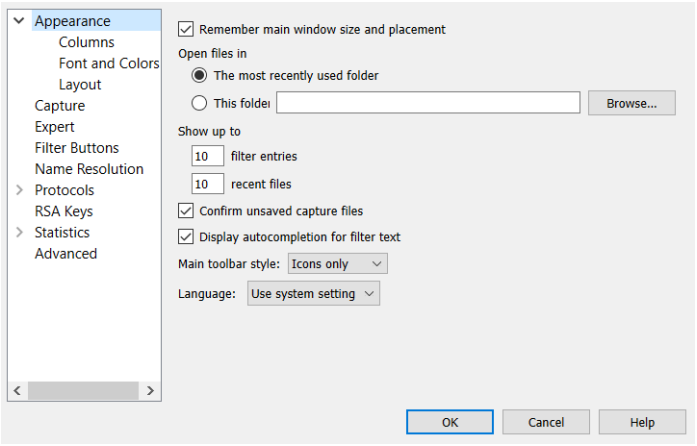

Figura 5.1: Configuración del Wireshark: ventana emergente

En la parte izquierda de la ventana, aparece > *Protocols*, que se trata de un menú desplegable. Una vez extendido, se debe buscar *DLT_ USER* y pulsar el botón *Edit* que nos abrirá la tabla mencionada anteriormente. Pulsando sobre el botón "+" se pueden añadir entradas a la tabla.

La Tabla 5.1 muestra los valores que se han de añadir para completar la configuración. Es importante destacar que solamente se muestran aquellos campos que se han de cambiar obligatoriamente:

Tabla 5.1: Entradas a añadir en la tabla DLT_USER de Wireshark

DLT	Payload Protocol
User 0 (DLT=147)	mac-lte-framed
User 1 (DLT=148)	lte-rrc.bcch.dl.sch
User 2 (DLT=149)	nas-eps
User 3 (DLT=150)	s1ap

ANEXO III: FORMATO DE LAS TRAZAS DE srsLTE

El software empleado permite realizar trazas tanto en la estación base como en el terminal móvil. Su formato y los campos que las constituyen se exponen a lo largo de este apartado.

La estructura de las trazas se puede dividir en tres partes: la primera se corresponde con las características de la señal recibida, la segunda recoge los parámetros del enlace DL y, por último, las características UL.

En el caso del UE las trazas están constituidas por estas partes pero, en el caso de la ENB solamente se aporta información sobre los parámetros DL y UL.

Las Tablas 5.2 y 5.3 muestran los campos que componen las métricas del UE y del ENB respectivamente:

Tabla 5.2: Formato de las trazas en el UE

Signal				**DL**						**UL**			
cc	rsrp	pl	cfo	mcs	snr	turbo	brate	bler	ta_us	mcs	buff	brate	bler

Tabla 5.3: Formato de las trazas en el ENB

DL						**UL**					
rnti	cqi	ri	mcs	brate	bler	snr	phr	mcs	brate	bler	bsr

A continuación, se resume la información que aporta cada una de las variables de la tabla anterior:

- **Component carrier (cc)**: Conjunto de frecuencias que se asignan a un determinado usuario para su comunicación.
- **Reference Signal Receive Power (rsrp)**: Potencia media de la señal recibida. Se mide en dBm.
- **Pathloss (pl)**: Atenuación sufrida por la señal causada ppor su propagación a lo largo del canal de comunicaciones. Se mide en dB.
- **Carrier Frequency Offset (cfo)**: Es el desplazamiento en frecuencia que sufre la portadora debido a la imposibilidad de sincronizar de forma perfecta el transmisor y receptor. En las trazas se mide en Hz.

- **Modulation and Code Scheme (mcs)**: Especifica el esquema de modulación y codificación empleado en la red. Su rango de valores (0-28) y su significado está estandarizado por el 3GPP. Se trata de una variable adimensional.
- **Signal-Noise Ratio (snr)**: Expresa la relación de potencias entre la señal recibida y su ruido. Indica cuánto se ha degradado la señal en el canal. Se mide en dB.
- **turbo**: Número medio de iteraciones que realiza el decodificador turbo (en inglés, turbo decoder).
- **Bitrate (brate)**: Tasa binaria a la que se está transmitiendo la información. Se mide en bits/segundo.
- **Block Error Rate (bler)**: Proporción de paquetes perdidos respecto al número total de paquetes transmitidos. Es adimensional.
- **Time Advance (ta_us)**: Valor indicado por el eNodeB al UE con el objetivo de compensar los tiempos de propagación de sus transmisiones y evitar que se produzcan colisiones entre las transmisiones de diferentes usuarios. Como la posición del móvil es variable, es necesario el envío constante de esta información. Además, permite al eNodeB gestionar el handover y permite la geoloocalización de los usuarios.
- **Buffer UL (buff)**: Cantidad de datos, medido en bytes, que están a la espera de ser transmitidos.
- **Radio Network Temporary Identifier (rnti)**: Identificador que permite diferenciar un canal radio del resto. Como cada canal es dedicado, permite diferenciar usuarios.
- **Rank Indicator (ri)**: Parámetro utilizado cuando se utilizan múltiples antenas para la transmisión y recepción. Indica el grado de interferencia entre las antenas (transmisoras o receptoras). Se mide en dB.
- **Channel Quality Indicator (cqi)**: Información sobre el enlace radio que envía el UE a la estación para determinar el esquema de modulación y codificación a emplear, es decir, el mcs.
- **Power Headroom (phr)**: Representa el margen de potencia existente entre la potencia a la que está transmitiendo el UE en ese instante y la máxima potencia a la que puede transmitir. Se mide en dB.
- **Buffer Status Report (bsr)**: Indica qué cantidad de información, medida en bytes, se encuentra a la espera de ser transmitida en el buffer del UE. Aporta la misma información que el campo *Buffer UL*.

ANEXO IV: INTRODUCCIÓN A LOS PRB

Los recursos de LTE tiene una estructura claramente definida. Cada trama tiene una duración fija: 10 ms y está formada por 10 subtramas (de 1ms de duración). Cada subtrama está constituida por 2 slots de duración 0.5ms.

Cada slot se mapea a un bloque físico de recursos o PRB. Cada bloque de recursos tiene una duración de 0.5ms y está formado por 12 subportadoras separadas entre sí 15kHz. Asimismo, estas subportadoras son ortogonales entre sí, por lo que, no se genera ninguna interferencia entre ellas.

$$\mathrm{BW}_{PRB} = 12 \text{ subportadoras} \cdot 15 \text{ kHz/subportadora} = 180 \text{ KHz}$$

Cada PRB está formado por 84 Resource Elements (RE), siendo un Resource Element una subportadora modulada en el tiempo de un símbolo [16]. Generalmente, en cada slot se envía 7 símbolos OFDM siendo su tiempo de símbolo (t_s) de 71 ns aproximadamente.

La Figura 5.2 resume la estructura de las tramas LTE y los elementos que la constituyen:

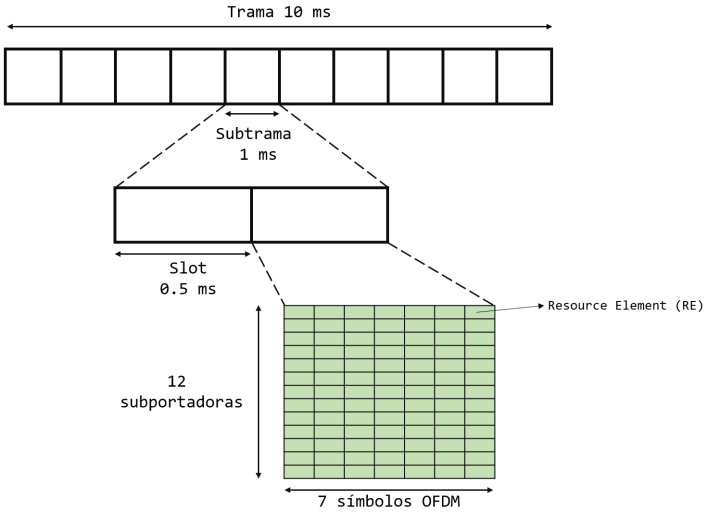

Figura 5.2: Estructura de las tramas LTE

El ancho de banda de operación está dividido en N recursos, cuyo valor varía en el siguiente intervalo: $6 \leq N \leq 100$. Al variar el número de PRB se consigue variar el ancho de banda ofrecido y permite adecuar la velocidad ofrecida en la red en funciones de las necesidades.

Por lo tanto, existe una correspondencia entre el número de PRBs y el BW ofrecido por la red. El reparto de recursos es una de las responsabilidades de *scheduling* que tiene el eNodeB. La Tabla 5.4 recoge la relación entre el BW y el número de PRBs:

Tabla 5.4: Relación PRB-BW

PRB	6	15	25	50	75	100
BW Ocupado (MHz)	1.08	2.7	4.5	9	13.5	18
BW Nominal (MHz)	1.4	3	5	10	15	20

En la tabla anterior se distinguen 2 tipos de ancho de banda: el de ocupación que se calcula con la fórmula mostrada a continuación y el BW nominal, que es el estandarizado:

$$\text{BW}_{Ocupado} = 180 \text{ kHz/PRB} \cdot \text{N PRB}$$

Además, se puede calcular el throughput máximo que se puede transferir, en condiciones ideales, para cada una de estas configuraciones. Para ello, es preciso tener en cuenta 2 factores: se emplea una eficiencia espectral de 5 bps/Hz y el 25 % de la tasa binaria se destina al envío de información de sobrecarga. Por lo tanto, el *throughput máximo* se calcula mediante la siguiente fórmula:

$$\text{Throughput}_{Máximo} = 0.75 \cdot \text{Eficiencia Espectral} \cdot \text{BW}_{Nominal}$$

A continuación, se resume en la Tabla 5.5 la relación entre el número de PRBs, el BW nominal y el throughput máximo:

Tabla 5.5: Relación PRB-BW-Throughput

PRB	6	15	25	50	75	100
BW Nominal (MHz)	1.4	3	5	10	15	20
Throughput (Mbps)	5.25	11.25	18.75	37.5	56.25	75

ANEXO V: IPERF: COMANDOS Y RESULTADOS

IPERF es una herramienta que permite generar flujos de datos con el objetivo de evaluar el rendimiento y la fiabilidad de una red. Trabaja con un modelo cliente-servidor. Para la realización de las pruebas se requieren 2 ordenadores donde uno actuará como cliente y el otro como servidor.

En primer lugar, es necesario establecer el servidor mediante el siguiente comando:

```
$ iperf3 -s
```

Al ejecutar este comando, el servidor se quedará escuchando, por defecto, en el puerto 5201.

Para iniciar el cliente, se ejecuta el siguiente comando:

```
$ iperf3 -c <IP_Servidor>
```

Por defecto, se establecerá una conexión TCP con el servidor. Si se requiere que el protocolo de capa de transporte sea UDP, se debe añadir la flag *-u*.

Asimismo, mediante el uso de las *flags* se pueden añadir parámetros a las pruebas, lo que permite evaluar la red en numerosos tipos de escenario. A continuación, se enumeran las flags que se han empleado en las pruebas sobre la red LTE desplegada:

- -i <tiempo>⟶ Parámetro utilizado para indicar cada cuántos segundos se quiere que se reporte información.
- -V ⟶ Aporta información adicional sobre los datos aportados por la herramienta. Por ejemplo, en el caso de UDP el jitter o, en TCP, el número de retransmisiones.
- -n <datos>⟶ Permite indicar cuántos datos se desean enviar. Es imprescindible indicar las unidades: k (kB), M (MB) o G (GB). Por ejemplo, si se desean mandar 20MB, se indicará del siguiente modo: *-n 20M*.
- -b <velocidad_fuente>⟶ Permite determinar la velocidad con la que IPERF genera los datos, es decir, la velocidad de fuente. Al igual que en el caso de la bandera anterior es preciso indicar las unidades: k (kbps), M (Mbps) o G (Gbps).

A continuación, se explica el intercambio el flujo de tramas entre el cliente y el servidor. Asimismo, se explica el formato de los datos obtenidos en cada caso:

Pruebas TCP

La Figura 5.3 muestra el intercambio de tramas que se produce. Primero se establece la conexión TCP y, posteriormente, se inicia el cliente comienza a enviar los datos al servidor, recibiendo por cada trama su ACK[1] correspondiente.

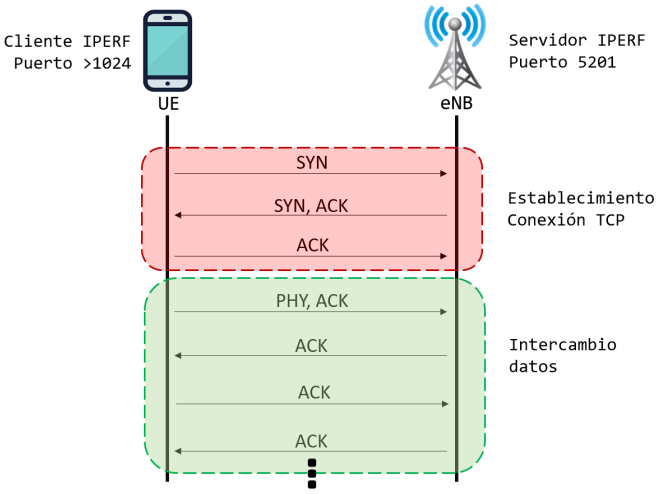

Figura 5.3: Intercambio de tramas empleando IPERF TCP

[1]Trama empleada para indicar que el paquete de datos al que va asociado ha sido recibido correctamente.

A continuación, se muestra en la Tabla 5.6 el formato de los resultados aportados por IPERF en este tipo de pruebas:

Tabla 5.6: Formato de las medidas IPERF empleando TCP

Id.	Interval	Transfer	Bandwidth	Retr.	Cwnd.

- Id: Permite identificar la transmisión.
- Interval: Franja temporal en la que se ha producido la transmisión sobre la que se aporta información.
- Transfer: Cantidad de datos enviados al otro extremo.
- Bandwidth: Velocidad media de transferencia a la que se han mandado los datos.
- Retr: Número de segmentos TCP retransmitidos en el intervalo.
- Cwnd: Cantidad de datos que puede mandar un extremo sin la necesidad de recibir confirmación por el recpetor de dicha información.

Pruebas UDP

Cuando se emplea el protocolo UDP no se establece una conexión cliente-servidor, por lo que, desde el inicio se transmite datos. Además, no se garantiza la correcta recepción de los paquetes de datos. Esto implica que el servidor no enviará un ACK para confirmar la correcta llegada del paquete enviado por el cliente. El diagrama de la Figura 5.4 muestra de forma gráfica esta explicación:

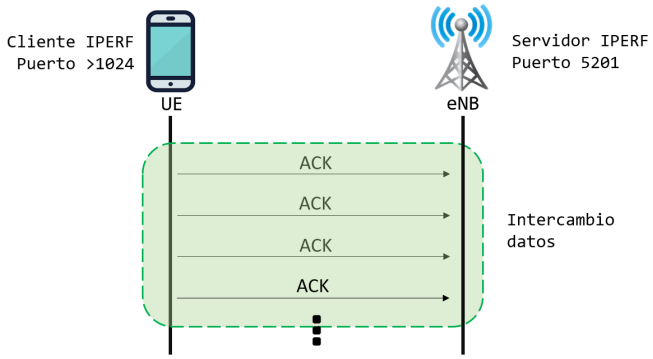

Figura 5.4: Flujo de tramas empleando IPERF UDP

A continuación, se muestra en la Tabla 5.7 el formato de los resultados aportados por IPERF en esta clase de evaluaciones de la red:

- Total Datagrams: Número total de datagramas enviados por el emisor durante el tramo temporal.

Tabla 5.7: Formato de las medidas IPERF empleando UDP

Id.	Interval	Transfer	Bandwidth	Total Datagrams

Además, una vez acabada la transmisión, IPERF aporta un resumen donde añade información adicional. La Tabla 5.8 muestra su formato:

Tabla 5.8: Parámetros del resumen de pruebas UDP

Id.	Interval	Transfer	Bandwidth	Jitter	Lost/Total Datagrams

- Jitter: Indica la variación media de la latencia en la prueba realizada.
- Lost/Total Datagrams: Porcentaje de paquetes perdidos respecto al total durante la transmisión.

Adicción de un timestamp a las medidas

Como se puede observar en las tablas anteriores donde se muestra el formato de los resultados que aporta IPERF, no existe ningún elemento que permite relacionar estos resultados con los aportados por *srsLTE*.

Por esta razón, se ha decidido añadir un sello temporal que resuelva este problema. El timestamp incluido es estático y representa cuándo se ha finalizado la prueba.

Sin embargo, este hecho no supone ningún tipo de inconveniente, ya que, conocidos el tiempo de finalización y la duración de la prueba se puede conocer de forma sencilla cuándo comenzó la prueba.

El método utilizado para conseguir el sello ha sido el siguiente, donde se ejecuta el comando *ts* que solicita al sistema operativo que informe acerca de la fecha y hora actual.

```
$ iperf3 -c <IP_Servidor> [optional-flags] | ts
```