UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

Despliegue y análisis de una red LTE basada en equipo SDR y srsLTE

TESIS
PARA OPTAR POR EL GRADO DE:
Maestro en Ciencia e Ingeniería de la Computación

PRESENTA:
Ing. Yoelkys Hernández Aracil

TUTOR PRINCIPAL
Dr. Víctor Rangel Licea

FACULTAD DE INGENIERÍA

CIUDAD DE MÉXICO, 10 DE ENERO DE 2020

Agradecimientos

Primeramente a la Universidad Nacional Autónoma de México y al Posgrado en Ciencia e Ingeniería de la Computación...
Por darme la gran oportunidad de superarme profesionalmente en este hermoso país que es México y contando con un excelente claustro de profesores. Me llevo, no solo un cambio profesional y el hecho de haber compartido con excelentes personas, sino también una marca de por vida: el orgullo de ser UNAM.

Al Consejo Nacional de Ciencia y Tecnología...
Por la ayuda económica otorgada para poder realizar esta Maestría.

A los proyectos...
- Papiit IN116316 “Diseño y evaluación de técnicas de calendarización aplicadas a los sistemas de transporte inteligente”.
- SECTEI No. 10507C19 “Desarrollo de infraestructura para la implementación de una red celular 5G utilizando un fronthaul basado en multiplexación por división de longitud de onda en fibra óptica”.
- “Predicción de Inundación utilizando Redes de Información de Emergencia Pluvial de detección en tiempo real con redes de telefonía móvil y WiFi” (EWIN).

A mi tutor...
Por haber confiado en mi y brindarme su guía y enseñanza en el desarrollo de este trabajo.

A mis amigos...
Por disfrutar conmigo de los buenos momentos y ayudarme en los malos. Sobre todo por haberme hecho soportable el comenzar desde cero en un país extranjero.

A mis padres y familia...
Por apoyarme siempre y mantenerse cerca, a pesar de la distancia.

A mi esposa e hijo...
Por darme tanto amor y cariño y ser el soporte de mi vida, mis dos grandes tesoros.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En los últimos años, las tecnologías de la información y la comunicación han evolucionado vertiginosamente y se ha extendido ampliamente su uso hasta jugar actualmente un papel importante en la vida de las personas. En este contexto, el Internet ha sido decisivo al permitir el acceso masivo a la información y facilitar la comunicación y compartición de contenido.

Se espera que para 2022, la cantidad de dispositivos conectados a redes basadas en Internet Protocol (IP) sea más de tres veces mayor que la población mundial (3,6 dispositivos en red per cápita) [1]. Uno de los aspectos que han detonado las cifras de conexiones y tráfico en Internet son las comunicaciones inalámbricas, especialmente las tecnologías móviles celulares. En este sentido, el tráfico de teléfonos inteligentes va superando al tráfico de PC ya que este último pasará del 41 % observado en 2018 a un 19 % del tráfico IP en 2022. Los teléfonos inteligentes representarán cerca del 50 por ciento del tráfico global de Internet para 2022, en comparación con el 23 por ciento en 2017, como se muestra en la figura 1.1.

Las comunicaciones móviles generalmente se dividen en generaciones:

- 1G representando las primeras comunicaciones analógicas,
- 2G donde se introducen los sistemas de transmisión digitales,
- 3G con las primeras transmisiones de datos en banda ancha,
- 4G donde se logra una red núcleo evoluciona basada en el protocolo IP y con soporte de banda ancha para ofrecer mayores tasas de transferencia de datos,
- y próximamente a desplegar, la 5G donde se apostará por frecuencias de portadora muy altas con anchos de banda masivos, mayor densidad de estaciones base y un mayor número de antenas [2].

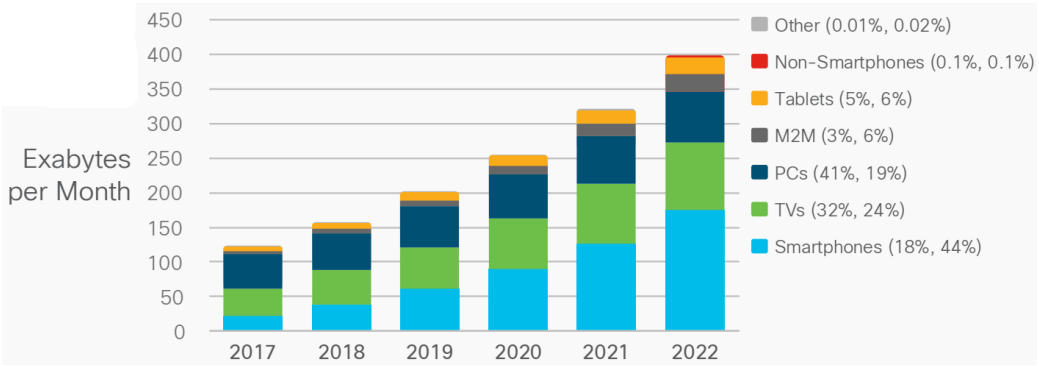


Figura 1.1: Tráfico global de IP por dispositivos [1].

*Las cifras entre paréntesis se refieren a: 2017, 2022.

Poder soportar servicios basados en el protocolo IP en un dispositivo móvil, que puede usarse ubicuamente con una conexión fija de banda ancha, ha sido todo un reto que se ha consumado con la llegada de Long Term Evolution (LTE) de 3GPP. LTE constituye la tecnología 4G predominante, donde no solo se integra la evolución de ciertos conceptos bien establecidos y consolidados en generaciones anteriores, sino que aparecen eficaces y novedosos fundamentos. Por esto, puede decirse que LTE en su estandarización como sistema para las comunicaciones móviles, combina evolución con revolución tecnológica.

1.1. Definición del problema

La alta demanda de servicios y contenido multimedia está desafiando la evolución de la tecnología 4G. Por esto se hace necesario buscar alternativas, no solo para aumentar la capacidad de las redes móviles actuales, sino de extender su alcance a zonas donde actualmente no existe ese servicio.

Satisfacer el incremento de tráfico, usuarios y servicios en un área determinada, sería posible mediante cambios regulatorios que permitan la asignación de mayor espectro radioeléctrico, o bien con el desarrollo y despliegue de nuevas tecnologías que mejoren la eficiencia espectral ofrecida. Aumentar el acceso al espectro normalmente es un proceso muy costoso y requiere de años dado su impacto legislativo. Esto deja una gran responsabilidad al estudio y diseño de nuevas técnicas que optimicen el uso del espectro disponible.

En cuanto a la accesibilidad, se presentan algunos inconvenientes respecto a las redes comerciales como es la estrecha asociación con hardware específico y falta de interfaces de control flexibles. Por lo que el despliegue de una red LTE convencional es un proceso muy costoso que los operadores, por cuestiones de rentabilidad, solo se ven tentados a ejecutar en lugares lo suficientemente poblados.

Para contrarrestar esta situación surgen algunos temas que han ganando interés y popularidad en el campo de las investigaciones relacionadas con las tecnologías inalámbricas como son los Software Defined Radio (SDR) y las Software Defined Networking (SDN). Este innovador paradigma propone desvincular las funciones de datos y de control. Abstrayendo este último en un plano lógico centralizado que simplifica grandemente la arquitectura, siendo los dispositivos de red (plano de datos) programados mediante software por determinada plataforma para simplemente procesar y transmitir paquetes. De esta forma, las redes se vuelven más rentables, controlables y flexibles [3].

SDR ha emergido como una eficiente tecnología de red capaz de soportar la naturaleza dinámica del futuro de las funciones de red y las aplicaciones inteligentes. A la vez que reduce los costos operativos simplificando el hardware y la administración y dándole más flexibilidad y simplicidad a la arquitectura.

A pesar de que los SDR son un área emergente y en pleno desarrollo, existen plataformas como srsLTE [4], de Software Radio Systems Company (SRS) [5], que proporcionan las pilas de protocolos L1, L2 y L3 tanto para UE como para eNodeB. Esta biblioteca de alto rendimiento, compatible de forma alineada con el Release 10 de LTE, es altamente modular con un mínimo de dependencias entre módulos o externas. Está disponible bajo licencias comerciales y de código abierto y utiliza una interfaz de comunicación compatible con los equipos Universal Software Radio Peripheral (USRP) de Ettus Research al usar la API USRP Hardware Driver (UHD) [6].

La biblioteca srsLTE proporciona las herramientas para construir aplicaciones basadas en LTE y equipos SDR, soportando la integración en forma modular de componentes de esta red como son: eNodeB, EPC y UE. Siendo posible, por ejemplo, obtener un LTE sniffer, analizar el rendimiento de la red; así como diseñar sistemas LTE para ambientes específicos, integrarlos con nuevas tecnologías y evaluar su comportamiento.

El presente trabajo está motivado por la importancia de comprender el funcionamiento de LTE con vistas a mejorar la eficiencia espectral de este estándar y sus sucesores. Así como el uso de la tecnología SDR como un método rentable y flexible con el que llevar y adaptar este servicio a zonas rurales donde actualmente no existe.

Para ello se centra en la obtención de una radio base LTE funcional, mediante el uso de equipos de radio definidos por software y la biblioteca de código abierto srsLTE, y en el estudio de su funcionamiento y desempeño.

1.2. Objetivos

1.2.1. Objetivo general

Desplegar y analizar el comportamiento de una red LTE basada en equipos de radios definidos por software y la plataforma srsLTE.

1.2.2. Objetivos específicos

- Entender el principio de operación de los equipos SDR de Ettus Research así como la API UHD para la comunicación con el nodo de procesamiento.
- Indagar en las tecnologías y conceptos involucrados en el funcionamiento del estándar LTE.
- Analizar el funcionamiento de la biblioteca srsLTE como plataforma LTE basada en equipos SDR.
- Describir los pasos para la instalación, configuración y despliegue de la plataforma srsLTE.
- Diseñar e implementar escenarios de prueba que permitan evaluar el desempeño del sistema en cuanto al comportamiento de la señal de radio, tasas de transmisión de datos y latencia del enlace.
- Caracterizar el desempeño del sistema a partir de datos obtenidos de las pruebas realizadas.

1.3. Metodología

Para cumplir con los objetivos de esta tesis, descritos anteriormente, se lleva a cabo su desarrollo en dos etapas: el estudio teórico y el desarrollo práctico. En el estudio teórico, el objetivo es revisar los conceptos teóricos pilares de la investigación.

Estudio del estándar LTE de 3GPP

- Investigación del estado actual de las redes LTE.
- Análisis de las funciones de los componentes de la arquitectura básica de una red LTE.
- Estudio de los conceptos implicados, principalmente, en el funcionamiento de las capas física y de enlace de datos en el estándar LTE.

- Indagación sobre los indicadores para la evaluación del desempeño de una red LTE en cuanto a la calidad de la señal de radio y las tasas de transmisión de datos.

Estudio de la plataforma SDR de Ettus Research

- Investigación del estado actual de la tecnología SDR.
- Estudio de la arquitectura y funcionamiento de SDR.

Estudio del software para la programación del sistema SDR

- Investigación sobre el funcionamiento y manejo de la plataforma srsLTE, así como su integración con los equipos SDR.
- Indagación sobre la instalación, configuración, adaptación y optimización de la plataforma srsLTE respecto a escenarios específicos de máximo desempeño de throughput.

En el **desarrollo práctico** la finalidad es obtener un modelo funcional del estándar LTE reconfigurable por software, por lo que la metodología a seguir es:

- Instalación y configuración del entorno sobre el que se va a desplegar la plataforma srsLTE.
- Verificación y validación del correcto funcionamiento del sistema obtenido.
- Implementación y despliegue de entornos de pruebas de desempeño del sistema.
- Captura, presentación y análisis de los resultados obtenidos para los indicadores de desempeño por cada entorno de prueba.

1.4. Relevancia y contribución

Con el presente trabajo se espera dar continuidad al estudio y caracterización de los SDR como tecnología emergente que pretende revolucionar y soportar a menores costos las crecientes demandas de tráfico y densidad en las comunicaciones inalámbricas.

Además, al obtenerse un despliegue funcional en software se espera motivar a futuras investigaciones e implementaciones que permitan optimizar el esquema obtenido, así como potenciar el desarrollo de otros estándares de redes inalámbricas de 4G y 5G.

Al documentar la implementación y análisis de una red LTE funcional con equipos SDR, se contribuye a proyectos de desarrollo e investigación en el área de comunicaciones inalámbricas.

1.5. Estructura de la tesis

El presente trabajo de investigación está dividido en siete capítulos. El Capítulo *2* muestra una introducción al estándar LTE y su estado actual en cuanto a las plataformas SDR que lo implementan. Además, aborda estudios relacionados con el despliegue y análisis de estas.

El Capítulo *3* define los conceptos más importantes del estándar LTE como arquitectura, pila de protocolos, formatos de tramas, señalización y esquemas de codificación y modulación, entre otros.

El Capítulo*4* aborda brevemente la historia del surgimiento de los SDR, los conceptos relacionados, descripción de su arquitectura, componentes y características. Así como los principales logros y potencialidades en el despliegue de una red LTE. Además, se describe el diseño y arquitectura de la herramienta srsLTE como plataforma LTE sobre tecnología.

El Capítulo *5* describe los aspectos del despliegue de srsLTE. Se tienen en cuenta los requerimientos de hardware y software, configuraciones, adaptaciones y pruebas iniciales de funcionamiento. También, se describen los escenarios de prueba diseñados para la caracterización del desempeño del sistema.

En el Capítulo *6* presenta la implementación de las pruebas de desempeño diseñadas, argumentándose la metodología a seguir y optimizaciones realizadas. Además, se muestran y analizan los resultados obtenidos en función de caracterizar el comportamiento de la red.

Finalmente, el Capítulo *7* expone las conclusiones de la investigación realizada, así como las recomendaciones y trabajos futuros acordes a este tema.

Capítulo 2

Estado del arte

El rápido crecimiento de la población de usuarios en redes móviles de las últimas décadas ha conducido al aumento exponencial en la demanda de conectividad y tráfico de datos en la red. Lo anterior está impulsado por la amplia gama de aplicaciones y servicios que se pueden consumir desde la mayoría de los dispositivos móviles, por citar algunos ejemplos: servicios de registro de ubicación, juegos online, vídeos en streaming, realidad virtual (VR) y realidad aumentada (RA), entre otros.

Para respaldar el crecimiento del consumo de aplicaciones y servicios desde y hacia las redes celulares, se necesita tener una cobertura ubicua de gran ancho de banda, como es la ofrecida actualmente por las redes de LTE. Dada su creciente popularidad se espera que estas redes de 4G puedan soportar en los próximos años el 58 % del total de las conexiones móviles sobre el 26 % observado en 2016 [7].

En el presente capítulo se hace un breve estudio de la historia evolutiva de las comunicaciones inalámbricas y del estado actual de LTE como tecnología de cuarta generación más utilizada universalmente. Además, se presentan y comparan las principales plataformas de software que actualmente implementan el protocolo LTE sobre equipos SDR. Exponiéndose por último un resumen sobre trabajos investigativos relacionados con la herramienta srsLTE, siendo esta la escogida para basar el desarrollo del presente trabajo.

2.1. Evolución de las comunicaciones móviles: logros y tendencias.

Tanto la telefonía fija como las primeras generaciones de la telefonía móvil fueron construidas para servicios basados en conmutación por circuitos, principalmente las llamadas de voz. El primer servicio de datos sobre GSM también estaba sobre esta tecnología y no fue hasta la llegada de GPRS cuando se comenzó a integrar la conmutación

por paquetes. Esto se heredó a los primeros desarrollos de tecnologías de 3G, aunque la conmutación por paquete y los servicios basados en IP se continuaban manejando como un complemento añadido.

El advenimiento de Internet para proveer todo tipo de servicios desde la década de los 90, se convirtió en una tendencia que ganaba fuerza. Los estándares de 2.5G y 3G soportaban algunos de los nuevos protocolos pero no estaban diseñados principalmente para consumir aplicaciones basadas en IP. Por lo que necesitaban evolucionar para hacer frente a la revolución de Internet.

El próximo paso era evidente, lograr una red de nueva generación basada en conmutación de paquetes y diseñada principalmente para soportar servicios y aplicaciones sobre el protocolo IP. Giro que se dio con la llegada de HSPA+ de 3G y luego, más concretamente, con LTE/LTE-Advanced.

2.1.1. Tecnologías de 4G

La primera versión de las especificaciones LTE, el Release 8, se completó en 2008 y la operación de la red comercial comenzó a fines de 2009. Paralelamente al desarrollo de LTE se dio una transformación de la arquitectura general de la red 3GPP. Denominada System Architecture Evolution (SAE), significó un rediseño de la red de acceso de radio plana con un solo tipo de nodo denominado eNodeB, así como una nueva arquitectura de red de núcleo llamada Evolved Packet Core (EPC) [8]. LTE a menudo se denomina `"4G"`, pero muchos afirman que el Release 10, también conocido como LTE-Advanced, es el verdadero paso hacia la 4G, siendo en ocasiones etiquetada la primera versión de LTE (Release 8) como 3.9G.

Combinando el poder de las tecnologías de acceso de radio de alta velocidad, con el poder de despliegue de aplicaciones innovadoras habilitado por Internet, el perfeccionamiento en la 4G de la red núcleo o EPC, es una piedra angular fundamental de la revolución de la banda ancha móvil. Diseñado además para garantizar una experiencia fluida tanto para los operadores como para los usuarios finales, ya que también brinda soporte para otras tecnologías de red de acceso de alta velocidad que no se basan en los especificaciones 3GPP (LTE, GSM y WCDMA/HSPA), por ejemplo, WiFi o acceso fijo.

La principal competencia de LTE intentó impulsarse desde IEEE con WiMax 802.16e y posteriormente 802.16m, como estándares de 4G. Aunque fue quedando desplazada eventualmente por LTE-Advanced. En la figura 2.1 se muestra un resumen del desarrollo de las generaciones móviles hasta 4G.

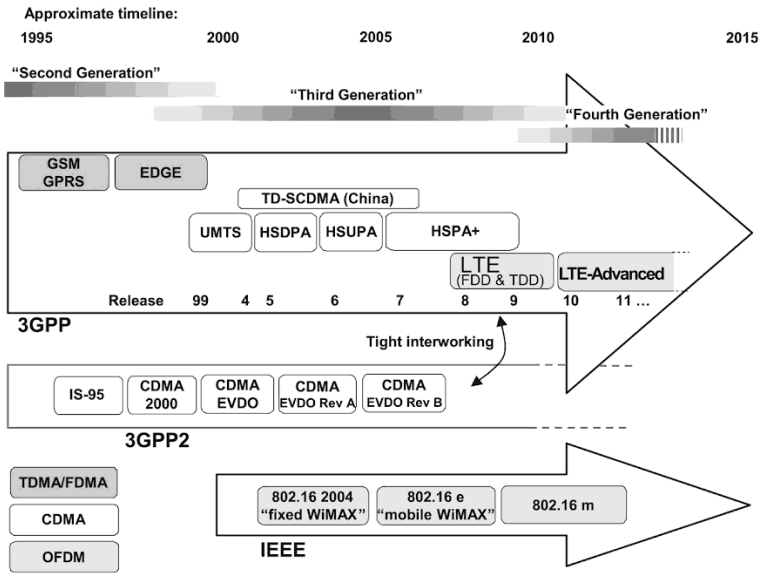


Figura 2.1: Línea de tiempo aproximada de las generaciones de comunicación móvil hasta 4G [9].

2.1.2. Tecnologías de 5G

La evolución futura a redes de 5G se propone abrir las puertas a nuevas oportunidades al ofrecer banda ancha móvil mejorada y velocidades de datos más rápidas y confiables con baja latencia en las comunicaciones. Impulsando campos como los vehículos autónomos, la cirugía remota y de forma general las comunicaciones inteligentes masivas de máquina a máquina (M2M) acarreadas por el fenómeno emergente del Internet of Things (IoT).

Uno de los aspectos más interesantes de las innovaciones de 5G es la capacidad para operar en bandas de frecuencias más altas y anchos de banda de canal más amplios. Se apuesta por las ondas milimétricas, con longitudes de onda más cortas y de mayor energía que las que puede manejar 4G [10].

2.2. Radios definidos por software SDR

En cuanto al diseño, desarrollo y despliegue de estándares de comunicación inalámbrica, se ha promovido el concepto de SDR por la flexibilidad y rentabilidad que representa. Entre sus definiciones más comunes está la de dispositivo de radio en la que algunas o todas las funciones de la capa física están definidas por software [11]. De esta forma, se puede reconocer a esta tecnología como un sistema de comunicación por radio

en el que los componentes que normalmente se definen en hardware se implementan mediante software en una computadora personal o un sistema integrado.

La gran evolución de las tecnologías de cómputo de mano con la emergente tecnología SDR abre una importante puerta para la mejora y expansión de los sistemas de comunicación inalámbrica.

Un aspecto interesante de los SDR es que al permitir la separación física de las funcionalidades puramente radio y de procesado digital. La unidad de radio podría estar distribuida en los emplazamientos desplegados sobre todo el territorio, mientras que la unidad digital podría ubicarse en un centro de gestión centralizado soportando multitud de emplazamientos unidos por un medio de Tx a altas velocidades como la fibra óptica [12].

Las ventajas mencionadas de estos sistemas motivan (dada su rentabilidad) que puedan utilizar para llevar sistemas inalámbricos de 4G como LTE a zonas de difícil acceso donde la cobertura de los operadores comerciales es escasa o nula. Además, al poder hacer uso de bibliotecas, disponibles en muchos casos en código abierto, con implementaciones de estándares inalámbricos, se hace posible diseñar y probar mejoras a los mismos sin necesidad de hardware específico. A partir de la investigación realizada, se presentan a continuación las principales plataformas SDR disponibles que implementan LTE.

2.2.1. openLTE

OpenLTE [13] es una implementación de código abierto escrita en C++ de la especificación LTE del 3GPP. Centra su desarrollo en la Tx y Rx del enlace descendente. Con esta es posible emular un eNodeB en una máquina con Linux [13], entre los requerimientos necesarios están:

- PC multicore con puerto USB 3.0 disponible y sistema operativo Linux.
- USRP B210.
- 2 antenas Vert900.
- UHD driver.
- GNU Radio 3.7.2 o superior.

El código de openLTE está bien organizado, documentado y es fácil de personalizar o modificar. Sin embargo, no está desarrollado completamente y muchas características están incompletas o son inestables. Además no proporciona una implementación de UE, lo que limita la realización de pruebas en términos de instrumentación y medición.

2.2.2. Arquitectura para aplicaciones LTE de LabVIEW

El framework de aplicaciones LTE de LabVIEW Communications [14] es un complemento de software que ofrece una implementación de la capa física LTE en tiempo real en forma de código fuente abierto y modificable, que soporta modos de transmisión TDD y FDD. Es compatible con un determinado subconjunto del Release 10 LTE del 3GPP, soportando una serie características que cumplen con la estructura principal del estándar LTE. Esta plataforma está pre-integrada y lista para ejecutarse en hardware de radio definida por software de NI.

Entre sus desventajas podemos mencionar los altos costos de las licencias que son necesarias adquirir al estar basado en software privativo. Además, en la actualidad es compatible solamente con el sistema operativo Windows (10/8.1/7 SP1) y con equipos USRP de National Instruments/Ettus USRP y PXIe, por lo que no se pueden utilizar, por ejemplo, dispositivos móviles convencionales para estudiar el comportamiento de la red.

2.2.3. OpenAirInterface (OAI)

OpenAirInterface (OAI), de OpenAirInterface Software Alliance (OSA), es una plataforma de tecnología inalámbrica basada en software de código abierto que ofrece una implementación de un subconjunto de la versión 10 de LTE para UE, eNB, MME, HSS, SGw y PGw en equipos informáticos de propósito general basados en Linux (arquitecturas Intel x86 PC/ARM). Está escrito en el lenguaje C y es distribuido libremente por la Alianza bajo los términos estipulados por el modelo de licencia OSA [15].

Puede usarse junto a equipos SDR disponibles en muchos laboratorios (es decir, las plataformas National Instruments/Ettus USRP y PXIe), además del hardware de RF personalizado proporcionado por EURECOM[16], para implementar estas funciones en un grado suficiente para permitir la interoperación en tiempo real con dispositivos comerciales.

Se puede usar para desplegar y personalizar una estación base LTE (eNodeB) y una red núcleo (EPC) en una PC, permitiendo además conectar dispositivos móviles (UE) y así monitorear el desempeño de diferentes configuraciones y diseños de red en tiempo real [17]. OAI es muy completo y proporciona un buen rendimiento. Sin embargo, una de sus desventajas es que la estructura del código es en cierto modo compleja y difícil de personalizar [18].

2.2.4. srsLTE

La biblioteca srsLTE para aplicaciones LTE basadas en SDR está escrita en el lenguaje C, es gratuita y de código abierto. Está desarrollada por SRS [19] y se lanza bajo

la licencia AGPLv3. Usa software del proyecto OpenLTE para algunas funciones de seguridad y para el análisis de mensajes NAS.

La biblioteca es compatible con los equipos RF que funcionan con los controladores de hardware UHD de Ettus o bladeRF, siendo necesario que cuenten con un reloj de 30.72 MHz para funcionar correctamente con frecuencias de muestreo LTE y señales de decodificación de estaciones base LTE en vivo. Es altamente modular con un mínimo de inter-módulos o dependencias externas. Incluye:

- srsUE: aplicación SDR de un UE LTE con todas las capas desde PHY a IP.
- srsENB: aplicación SDR de un eNodeB LTE.
- srsEPC: implementación ligera de la red núcleo LTE que incluye representaciones del MME, HSS y SGW/PGW.

Las plataformas OAI y srsLTE resaltan de entre las anteriormente mencionadas dado que ambas son de código abierto y demuestran estabilidad y buen rendimiento en tiempo real. Cuentan con una gran comunidad activa de desarrollo y documentación. Sin embargo, algunos trabajos como [20] y [21] destacan el desempeño srsLTE respecto OAI en cuanto a términos de estabilidad, tiempos de ejecución y ocupación de la memoria. Para basar el desarrollo del presente trabajo se decide utilizar srsLTE dada su gran modularidad y alto rendimiento.

2.3. Trabajos relacionados

En esta sección, se presentan algunos trabajos que abordan la herramienta srsLTE para realizar estudios de su desempeño o bien del estándar LTE.

En [18] se presenta la biblioteca srsLTE como una útil herramienta para el estudio, experimentación e innovación orientada a LTE. Se describen sus componentes de software acentuando su modularidad, eficiencia computacional y la idoneidad para investigar sobre futuras mejoras para LTE. En este sentido se presenta un caso de estudio donde se realizan modificaciones a la biblioteca para la convivencia de LTE y WiFi en bandas no licenciadas. El módulo srsUE, capaz de emular un equipo de usuario, es probado usando un USRP B210 y una PC con CPU Intel Core i7-4790. Muestran resultados de más de 60 Mbps en el enlace descendente con una configuración SISO [1] y ancho de banda de 20 MHz.

En [20] se presentan algunas plataformas existentes de SDR basadas en GPP, centrándose posteriormente en OAI y srsLTE. Su propósito es demostrar el buen desempeño del enfoque GPP a través de una comparación de estas. Para esto se analiza

[1] Se utiliza solo una antena para la Rx y otra para la Tx.

el código de ambas y se realizan pruebas de rendimiento a la capa física del enlace descendente. Para realizar las pruebas utilizaron dos PCs con similares características equipadas con Intel(R) Core(TM) i7-4770 CPU de cuatro núcleos físicos y frecuencia máxima de 3.4 GHz. Como sistema operativo empleó Ubuntu 14.04 de 64-bit y emplearon el USRP B210. Los escenarios de prueba se diseñaron con el objetivo de evaluar ambas plataformas teniendo en cuenta el throughput máximo alcanzado en función del MCS y el rendimiento computacional de las implementaciones de capa física para el downlink. Los resultados muestran, en el caso de srsLTE, un throughput máximo de 12.6603 Mbps y 24.7209 Mbps para los anchos de banda de 5 10MHz respectivamente, usando un MCS de 27.

Por su parte, en [22] se realiza una análisis de la herramienta srsLTE y se despliegan los módulos necesarios para la Tx y Rx de datos usando LTE. Como interfaz RF del eNB y UE se utilizan USRPs B210 conectados a PCs con CPU i5 y SO Ubuntu 12.04. Se hacen pruebas usando diferentes esquemas de modulación (BPSK, QPSK, 16-QAM y 64-QAM) y son registrados los parámetros SNR (relación señal/ruido), BLER (tasa de error de bloque) y CFO (compensación de frecuencia de portadora).

Como parte de la investigación realizada en [21], se hace un despliegue y valoración de la plataforma srsLTE en su versión 18.3.1. Para hostear el eNB se utiliza una PC con 8 núcleos de CPU a 3.4 GHz, 16 GB de RAM y SO Ubuntu 14.04.5. Como equipo RF emplean el NI USRP-2942R [1] conectado a la PC mediante la interfaz PCI Express de alta velocidad y el equipo de usuario utilizado fue un celular Motorola G5 Plus con una tarjeta USIM personalizada.

Según muestran los resultados, la conexión entre el eNB y el UE se mantuvo estable por más de una hora para los anchos de banda de 5 MHz y 10 MHz. Se realizaron pruebas de throughput (Speed Test) obteniéndose para los anchos de banda 5 MHz, 10 MHz y 20 MHz los valores: 15 Mbps, 22 Mbps y 23 Mbps para DL y 9 Mbps, 13 Mbps y 22 Mbps para UL; respectivamente.

[1] Posee hardware equivalente al X310 con 2 tarjetas hija SBX y soporte para la plataforma LabView.

Capítulo 3

Long-Term Evolution (LTE)

En el presente capítulo, se describe la arquitectura de una red de comunicaciones móviles basada en las especificaciones del estándar LTE del 3GPP. Para ello se identifican los componentes de alto nivel y se realiza una descripción de ellos en base a las entidades de red e interfaces asociadas. De las interfaces se describen sus funciones y la pila de protocolos que las sustentan. También se tratan los esquemas de transmisión, la programación de paquetes, el soporte de múltiples antenas y la flexibilidad del espectro, entre otros temas de interés. Se proporciona una descripción general de la estructura del protocolo, incluidos RLC, MAC y la capa física, explicando los canales lógicos y físicos, y el flujo de datos relacionados.

Por último se ofrece y argumentan una serie de métricas que permiten valorar la calidad del canal de radio y por ende del buen desempeño de la red. Estas son utilizadas como base para la caracterización del sistema LTE cuyo despliegue e implementación de pruebas se expone en próximos capítulos.

3.1. Introducción

El término de LTE fue acuñado en los inicios de sus investigaciones por el 3GPP para denominar una línea interna de trabajo cuyo objetivo era la evolución de la red UMTS. Concerniente a estas investigaciones, la red de acceso se denominó como Evolved UTRAN (E-UTRAN) y los términos SAE y EPC fueron utilizados para referirse a las actividades de estudio relacionadas con la red troncal evolucionada completamente basada en conmutación de paquetes. La combinación de la red de acceso E-UTRAN y la red núcleo EPC fue introducida en el Release 8 de las especificaciones del 3GPP. Esta constituye la nueva red UMTS evolucionada y recibe el nombre de Evolved Packet System (EPS). Dado su uso comercial, es común encontrar el término LTE como sinónimo de EPS, en el presente trabajo se adopta igualmente este criterio. A su vez, las descripciones ofrecidas se centran en el modo FDD y el Release 10 de LTE para alinear la compatibilidad con la plataforma srsLTE.

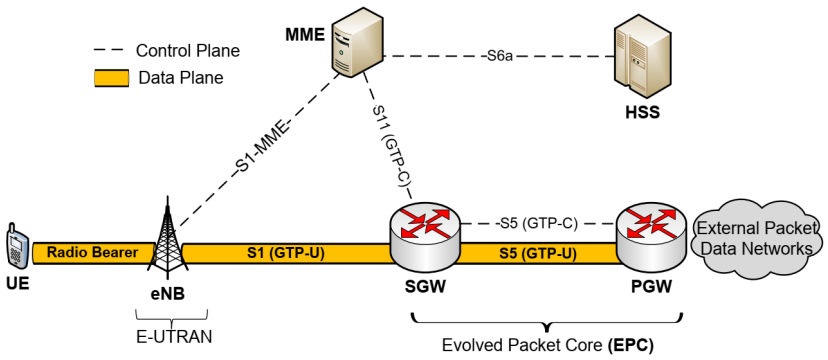


Figura 3.1: Arquitectura básica para LTE [23].

Entre las características generales de LTE están el uso de canales de frecuencia con anchos de banda variables para posibilitar mayor compatibilidad con tecnologías anteriores: 1.4, 3, 5, 10, 15, y 20 MHz. Como esquema de acceso en la capa física se utiliza Orthogonal Frequency Division Multiple Access (OFDMA) para el enlace descendente y Single Carrier Frequency Division Multiple Access (SC-FDMA) para el enlace ascendente, así como los modos de transmisión Frequency División Duplexing (FDD) y Time Division Duplex (TDD).

Se integran métodos de modulación y codificación adaptativa así como algoritmos avanzados para la asignación de recursos de forma dinámica y eficiente. Además, se utilizan esquemas de modulación de orden superior, como 16-QAM y 64-QAM y soluciones avanzadas de múltiple antena (MIMO), que incluyen diversidad de transmisores y receptores, formación de haces (beamforming) y multiplexación espacial.

3.2. Arquitectura LTE

Como muestra la figura 3.1, la arquitectura LTE tiene componentes elementales requeridos para admitir la conectividad IP básica a través del acceso LTE. Estos componentes son: equipo de usuario (UE), red de acceso (E-UTRAN) y red troncal (EPC). La arquitectura completa del EPS en la que se sostienen los sistemas LTE comerciales es mucho más compleja ya que incluye elementos de las generaciones celulares anteriores para propiciar la interoperabilidad.

El diseño mostrado de la arquitectura está guiado mayormente por dos principios [24]. El primero se refiere a la importancia de optimizar el tráfico de datos del usuario reduciendo su paso por nodos de procesamiento. Esta optimización, está motivada no solo por la disminución de la latencia en las comunicaciones, sino también por la

búsqueda de rentabilidad en futuros escalamientos de la arquitectura. Un argumento importante si tenemos en cuenta el acelerado crecimiento en el uso de datos móviles.

El otro principio importante concierne a la separación del plano de control (mostrado como línea discontinua en la figura 3.1) al plano de datos del usuario. Se hace necesario separarlos debido a que el tráfico de señalización escala diferente (dependiente del número de usuarios) al tráfico de datos de usuario (dependiente, por ejemplo, de la cantidad y tipo de servicios en ejecución por parte de los usuarios). De esta forma, teniendo en cuenta el primer principio, se puede optimizar el tráfico de datos. Esta separación lógica, propiciada por el incremento significativo de capacidad de computación de los equipos, permite además una mayor flexibilidad en el despliegue de la red. Siendo posible ahorrar recursos al ubicar de forma centralizada una infraestructura de manejo de señalización de control y distribuir, por su parte, los equipos de acceso a los usuarios.

Cabe destacar que las diferentes interconexiones física que se utilizan entre los equipos de una red LTE, también conocida como red de transporte, es una red IP convencional. Por lo que se pueden emplear rúters, servidores Dynamic Host Configuration Protocol (DHCP) para la asignación automática de direcciones IP a los equipos de la red LTE y servidores Domain Name Server (DNS) para asociar los nombres de dominio con sus direcciones IP.

3.2.1. E-UTRAN

Como se muestra en la figura 3.2, la E-UTRAN está compuesta por eNBs que brindan conectividad entre los equipos de usuario y la red troncal. Las interfaces lógicas de comunicación que utilizan para esto son:

- Uu: representa el enlace de radio directo que permite la transferencia de información entre el eNB y el UE.
- S1: es la interfaz de conexión al EPC que realmente está dividida en dos: S1-MME y S1-U. S1-MME comunica el eNB con el MME para soportar las comunicaciones del plano de control, mientras S1-U conecta con el SGW para el envío de tráfico del plano de datos del usuario.
- X2: interfaz opcional que comunica varios eNB entre sí para intercambiar, tanto información de control para la gestión de recursos (por ejemplo, para evitar interferencias), como tráfico de datos cuando se realiza el handover[1].

[1]Proceso mediante el que se transfiere el servicio de un eNB a otro cuando la calidad del enlace es insuficiente en uno de ellos, generalmente producto a la movilidad del usuario

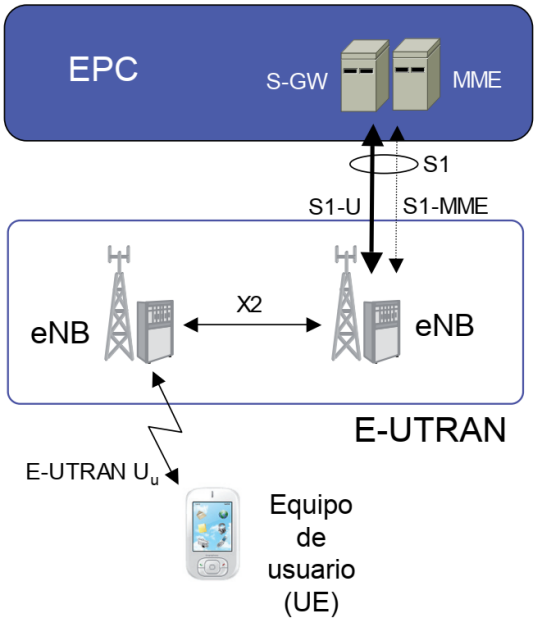


Figura 3.2: Red de acceso E-UTRAN [12].

3.2.1.1. eNB

Entre los componentes de la arquitectura mostrada en la figura 3.1 encontramos al menos un eNB (estación base LTE). Su responsabilidad asumir las funciones necesarias para realizar las conexiones inalámbricas reales entre los dispositivos del usuario y la red. Ejecuta y gestiona los protocolos de la interfaz de radio, tanto del plano de usuario como del plano de control.

En un escenario de red convencional puede haber varios miles de eNBs, muchos de estos pueden interconectarse a través de la interfaz X2 para permitir eficientes handovers. Realizan de la asignación dinámica de recursos a los usuarios (packet scheduling) considerando el Channel Quality Indicator (CQI), reportado por los móviles. Además, agrega mecanismos HARQ que combinan la corrección de errores en recepción a través de la incorporación de redundancia (FEC), con la retransmisión de paquetes (ARQ) a partir de códigos detectores de errores [12]. Otro de los mecanismos es el de soft-combining mediante el cual se aprovecha la parte útil o no dañada de los paquetes y sus retransmisiones para construir un único paquete más fiable que sus constituyentes.

3.2.2. EPC

La arquitectura evolucionada del dominio de conmutación de paquetes únicamente, consiste en varios tipos diferentes de nodos lógicos. Sin embargo, en una implementa-

ción física funcional, varios de ellos pueden combinarse, por ejemplo, MME, P-GW y S-GW podrían muy bien formar un único nodo físico [8].

A continuación se describen los principales componentes del EPC, aunque cabe destacar, que puede incluir otros tipos de nodos con funciones más específicas como el Policy and Charging Rules Function (PCRF), responsable del manejo y cobro de la calidad de servicio (QoS), o Multimedia Broadcast Multicast Services (MBMS), para dar soporte a la red.

3.2.2.1. Mobility Management Entity (MME)

Los eNB están conectados al menos a un MME a través de la interfaz lógica S1-MME. La principal responsabilidad de este último es administrar el plano de control de los equipos de usuario conectados a la red LTE mediante los protocolos Non-Access Stratum (NAS) [1] y controlar el tráfico del plano de usuario a través de la interfaz S11 con el SGW.

Sus funciones incluyen la conexión/liberación de servicios portadores, gestión de movilidad y seguridad para dispositivos y terminales. La MME administra todos los terminales en modo inactivo (idle mode), incluido el soporte para la gestión del área de seguimiento Tracking Area Management (TAM) y el aviso para reactivar la conexión (Paging). Para comunicarse entre ellas, las entidades MME utilizan la interfaz S10.

3.2.2.2. Home Subscriber Server (HSS)

Para realizar sus funciones, el MME se conecta a través de la interfaz S6a a una base de datos con registros de los usuarios autorizados a conectarse y la configuración de las correspondientes subscripciones. Este elemento se conoce como HSS. Los datos de suscripción que se guardan pueden incluir credenciales para autenticación y autorización de acceso y elementos para la gestión de movilidad entre redes LTE o entre esta y otras redes de acceso.

3.2.2.3. Serving Gateway (SGW) y Packet Data Network Gateway (PGW)

El flujo de datos de usuarios que fluye desde y hacia los dispositivos móviles se gestiona mediante los nodos lógicos llamados SGW y PGW. Para la interconexión entre ellos se utilizan las interfaces: S5, en caso de que el usuario esté en dominios de su propia red, o S8, si el usuario está en una red LTE visitada consumiendo roaming.

[1] Protocolos para el intercambio de señalización entre el equipo de usuario y la red troncal.

El SGW juega un papel muy secundario en las funciones de control. Solo es responsable de sus propios recursos y los asigna en función de solicitudes de otras entidades de red, como MME, PDN-GW o PCRF, que a su vez actúan sobre la necesidad de configurar, modificar o borrar portadores para el UE. Además, actúa como un anclaje de movilidad local cuando los terminales se mueven entre los eNB.

Cuando un UE está inactivo, el SGW almacena en un buffer los paquetes dirigidos a él que son entregados por el PGW y solicita al MME que inicie la búsqueda y llamado del UE. Si el Paging da resultado el UE se vuelve a conectar, y una vez restablecidos los túneles, los paquetes almacenados en el buffer se enviarán. Entre sus funciones también está la de recopilar información necesaria para la contabilidad y facturación de los datos de usuario.

Por su parte el PGW funciona como un rúter de borde entre el EPS y las redes externas de paquetes de datos que dan acceso a Internet. Realiza funciones de filtrado de paquetes según lo exijan las políticas establecidas para el UE y el servicio en cuestión, y recopila e informa los datos de facturación relacionados. Frecuentemente asigna la dirección IP al UE cuando este se conecta a la red y solicita una conexión Packet Data Network (PDN). Para esto el PGW puede funcionar como servidor DHCP o bien consultar un servidor DHCP externo.

Además, puede configurar los servicios portadores según la solicitud, ya sea a través de la PCRF o desde el SGW, proveniente del MME. En el último caso, también puede necesitar interactuar con la PCRF para recibir la información de control de política apropiada, si eso no está configurado localmente en el PGW. Funciona como ancla de movilidad para tecnologías de acceso de radio que no son 3GPP, como CDMA2000.

3.2.3. User Equipment (UE)

Los UE son los dispositivo que permite al usuario acceder a los servicios de la red LTE a través de una interfaz radio. Pueden incluir dos elementos básicos:

1. Tarjeta inteligente (Universal Integrated Circuit Card (UICC)) , también llamada Universal SIM (USIM) en LTE, conteniendo el International Mobile Subscriber Identity (IMSI) del usuario, su clave secreta asociada, entre otros parámetros necesario para permitir la conexión a la red y la utilización de sus servicios.

2. Equipo móvil en sí, denotado como Mobile Equipment (ME).

La separación entre USIM y ME facilita que un usuario pueda cambiar de terminal manteniendo su identidad. Con el objetivo de introducir un cierto grado de flexibilidad en el diseño del UE, se define una interfaz que permite que exista una separación física entre el equipo que alberga las funciones propias de la comunicación (MT) y el equipo que se ocupa de la interacción con el usuario (TE). La misma permite el uso de un

conjunto de comandos AT que permiten acceder a los servicios de la red (por ejemplo, establecimiento de una conexión) soportados en el MT desde el TE [25].

3.2.3.1. Categoría UE

Los UE se clasifican en un conjunto de categorías definidas por 3GPP (UE-Category), de las cuales se muestra un resumen en la tabla 3.1, basado en las velocidades máximas de datos y las compatibilidades MIMO del dispositivo.

Estas categorías se establecen en función de las capacidades en la Tx/Rx del terminal para soportar mecanismos de multiplexación espacial con múltiples antenas, el uso de determinadas modulaciones, el tamaño en bytes de las colas de transmisión, etc. Por lo tanto, como parte de la configuración de la conexión, el terminal indica no solo qué versión de LTE admite, sino también, según sus capacidades de capa física, a qué categoría UE pertenece.

Categoría	Downlink		Uplink	
	Velocidad máxima aprox. (Mb/s)[1]	Antenas para MIMO[2]	Velocidad máxima aprox. (Mb/s)	64QAM
1	10.296	1	5.16	No
2	51.024	2	25.456	No
3	102.048	2	51.024	No
4	150.752	2	51.024	No
5	299.552	4	75.376	Sí
6[3]	301.504	2	51.024	No
7[3]	301.504	2	102.048	No
8[4]	2998.560	8	1497.760	Sí

Tabla 3.1: Categorías de UE para Release 10 de LTE según 3GPP TS 36.306 [26].

[1]Al utilizarse MIMO en el enlace descendente varios Transmission Time Interval (TTI) de 1ms pueden transmitirse en paralelo.

[2]Modo de multiplexación espacial.

[3]Un UE que indique las categorías 6 ó 7 también indicará la categoría 4.

[4]Un UE que indique la categoría 8 también indicará la categoría 5.

Los valores teóricos de máximo throughput mostrados en la tabla consideran la utilización del formato de modulación de 64QAM en el DL y sin los atenuantes añadido por los efectos de la propagación, el acceso multiusuario y los mecanismos de retransmisión.

Los terminales móviles de categoría 1 a 4 utilizan un formato modulación de hasta 16 QAM mientras que los terminales móviles de categoría 5 utilizan el formato de modulación 64 QAM.

3.2.4. Numeración, direccionamiento e identificación

Para el correcto funcionamiento de la red LTE, los componentes físicos están relacionados con identificadores lógicos [27] como se muestra en la figura 3.3. Al igual que en generaciones anteriores, cada red en sí se identifica con un Public Land Mobile Network Identity (PLMN-ID). Este número está compuesto por: tres dígitos del código del país o Mobile Country Code (MCC) y dos dígitos del código de red o Mobile Network Code (MNC).

Varios MME pueden estar agrupados en una área lógica que se identifica con el MME Group Identity (MMEGI) compuesto por 16 bits. Dentro de una misma área, a cada MME le corresponde un código de 8 bit llamado MME Code (MMEC). De esta forma al combinar los anteriores, se forma el MMEI para identificar un único MME dentro de una red. A su vez, cuando se combina el PLMN-ID con el MMEI se forma el Globally Unique Mobility Management Entity Identifier (GUMMEI) para identificar únicamente y de forma global al MME.

Al igual que los MME, existen áreas de agrupación para los servicios de S-GW y ambas están compuestas por unidades más pequeñas y no superpuestas conocidas como áreas de seguimiento o Tracking Areas (TA) [28]. El objetivo de estas es mantener un seguimiento de la posible ubicación de los móviles que se encuentran registrados, o sea en estado EMM-Registered (ver sección 3.3), pero no tengan una conexión establecida con alguna estación base, o sea que se encuentran en estado de espera o ECM-Idle. Para identificar de forma global estas áreas de seguimiento se utiliza el Tracking Area Identity (TAI), código compuesto por 16 bits del Tracking Area Code (TAC) y el identificador de red.

Por su parte cada celda está identificada de forma local con un código de 28 bits denominado E-UTRAN Cell Identity (ECI) y de forma global con el E-UTRAN Cell

Global Identifier (ECGI). Otro de los identificadores importantes en este caso es el Physical Cell Identity (PCI) consistente en un número de 0 a 503 para distinguir la celda de sus vecinas inmediatas.

Como se describe en el apartado anterior, el equipo de usuario debe poseer una tarjeta USIM que está identificada por el IMSI, y a su vez el propio equipo posee un identificador único denominado International Mobile Equipment Identity (IMEI). Estos códigos, principalmente el IMSI, pueden ser usado para suplantar la identidad de un suscriptor de la red. Por esto es importante evitar, siempre que sea posible, el envío de esta información por la interfaz de aire. Para lograr que un equipo ya registrado no se identifique en la red usando el IMSI se utilizan identificadores temporales, por ejemplo, el Globally Unique Temporary Identity (GUTI) y el Temporary Mobile Subscriber Identity (TMSI) que identifican al suscriptor de forma global y local, respectivamente.

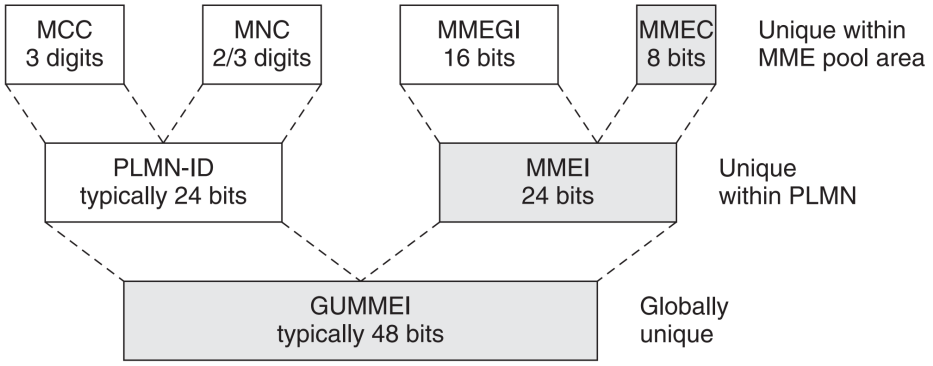


Figura 3.3: Identificadores usados por el MME [28].

3.3. Gestión de servicios portadores EPS

Como se ha visto anteriormente, la red núcleo de LTE funciona completamente sobre el protocolo IP lo que la hace más compatible con Internet y las aplicaciones que ahí viven. Sin embargo, existen algunos elementos que siguen diferenciando una red LTE de una red convencional de equipos conectados a Internet. De estos elementos, se puede decir que los principales a tener en cuenta son la movilidad (ver sección 3.4) y la calidad de servicio. En el primero de los casos se refiere a que los equipos UE en una red LTE pueden cambiar su ubicación al ser móviles.

En cuento a la calidad de servicio (QoS) es un factor que no ofrece Internet por su mecanismo imparcial de entrega de mejor esfuerzo. Sin embargo, en LTE es necesario realizar una diferenciación para ofrecer cierta garantía de servicio, por ejemplo, al tráfico VoIP para llamadas de voz o vídeo en tiempo real. Además, con un enfoque más comercial, dependiendo del tipo de usuario y el costo de su plan, debería poder ofrecerse

mayor calidad en la conexión o mayor prioridad a algunos servicios. Para poder hacer posible esta diferenciación de QoS entre usuarios y/o servicios se utilizan los servicios portadores EPS.

Estos servicios funcionan como conductos o tuberías bidireccionales que transfieren los paquetes IP a través de la red LTE usando determinados parámetros de QoS y teniendo en cuenta la plantilla de flujo o filtro de paquetes conocida como Traffic Flow Template (TFT).

Para que el UE tenga conectividad a la red de conmutación de paquetes en LTE, y salida a Internet, es necesario que se haya activado al menos un servicio portador. Cuando se está estableciendo una nueva conexión se establece un servicio portador por defecto por el cual se enviará el tráfico IP del usuario sin distinción alguna. Posteriormente, si el usuario utiliza aplicaciones que requieren de mayor garantía del servicio (por ejemplo, una videoconferencia) se pueden activar servicios portadores EPS dedicados. En este caso el TFT es el encargado de filtrar el tráfico para enviarlo a través de servicios portadores dedicados o el servicio portador por defecto.

El servicio portador por defecto suele estar activo durante todo el transcurso de la conexión PDN. Mientras que los servicios dedicados suelen activarse, modificarse o desactivarse a demanda, generalmente vinculado a una aplicación final que requiere un tratamiento especial en la calidad de la conexión. Cabe destacar que los servicios portadores EPS relacionados a una misma conexión PDN, comparten los mismos parámetros de conectividad IP. Un equipo móvil puede tener asignado hasta 11 servicios portadores EPS para darle conectividad a varias redes utilizando diferentes parámetros de QoS [29].

3.4. Gestión de movilidad

Respecto a la gestión de movilidad, constituye uno de los requisitos claves que caracterizan los sistemas móviles. Se puede decir que satisfacer los requisitos de movilidad incluye principalmente dos aspectos. Primero, que el equipo de usuario pueda ser localizado, avisado y le puedan ser entregados paquetes originados en la red y que vayan dirigido hacia este, siempre que se encuentre bajo cobertura y aun estando en modo inactivo. Y lo segundo, que cuando el usuario tenga conexiones activas y se encuentra en movimiento, pueda moverse de una celda a otra de la red sin perder dicha conexión (handover).

Para gestionar la movilidad es necesario poder determinar cuándo el usuario se encuentra registrado en el EPC y cuándo está en modo activo o inactivo. Para modelar este comportamiento se utilizan diagramas de estado como se muestra en la figura 3.4.

El primero de los diagramas representa un modelo de movilidad denominado EPS Mobility Management (EMM) con dos posibles estados que describen si el UE se encuentra o no registrado en los servicios del EPC. En el estado EMM-REGISTERED el móvil se encuentra encendido y está dado de alta en los registros del MME y el SGW, siendo posible realizar su localización a través de áreas de seguimiento o Tracking Area list (TAL). Posee una dirección IP válida y el servicio portador EPS por defecto, dándole conectividad PDN.

Por su parte, el estado EMM-DEREGISTERED se alcanza cuando no existe información de localización del móvil, debido generalmente a que este ha sido apagado o salió del área de cobertura. Para pasar de un estado a otro se llevan a cabo los procesos de registro o attach (ver sección 3.4.1) y cancelación de registro o detach.

La figura 3.4 muestra los estados definidos para indicar la existencia o no de un plano de control activo entre el UE y el MME donde se encuentra registrado, este modelo se conoce como EPS Connection Management (ECM). En este contexto el UE tiene dos estados: RRC-CONNECTED (o conectado) o RRC-IDLE (o inactivo). En el primer estado, el UE está activo y con un contexto RRC (ver sección 3.5.1) por lo que se puede realizar el envío y recepción de datos.

En el segundo estado, el UE no tiene sesiones activas en ninguna celda y se liberan los recursos de radio y el plano de datos. En este estado el móvil puede ahorrar batería ya que, al no tener asignados recursos de radio para transmitir, solo escucha frecuentemente los mensajes de difusión enviados por la radio base. Si el móvil sale de su TAL tendrá que notificar al MME, de esta forma en caso de que este último requiera contactar con el UE, puede hacerlo enviando un mensaje de llamado a las estaciones base dentro de las áreas de seguimiento definidas.

3.4.1. Proceso de registro del usuario en la red LTE

Una vez encendido, el primer paso que debe realizar el UE es sincronizarse a la estación base. Para lograrlo debe recibir y decodificar las señales de sincronización primaria y secundaria (PSS y SSS) que permiten reconocer y delimitar las transmisiones de slots y tramas provenientes del eNB [31]. Así como compensar las diferencias en frecuencia y fase entre los osciladores del eNB y el UE.

Posteriormente, el equipo de usuario debe obtener información importante contenida en los mensajes MIB y SIB provenientes de la estación base. Estos mensajes son transmitidos en claro frecuentemente por el eNB conteniendo información de la red LTE, necesaria para el proceso de attach.

Luego ocurre el procedimiento de acceso aleatorio para establecer una conexión RRC con el eNB utilizando un Random Access Radio Network Temporary Identity

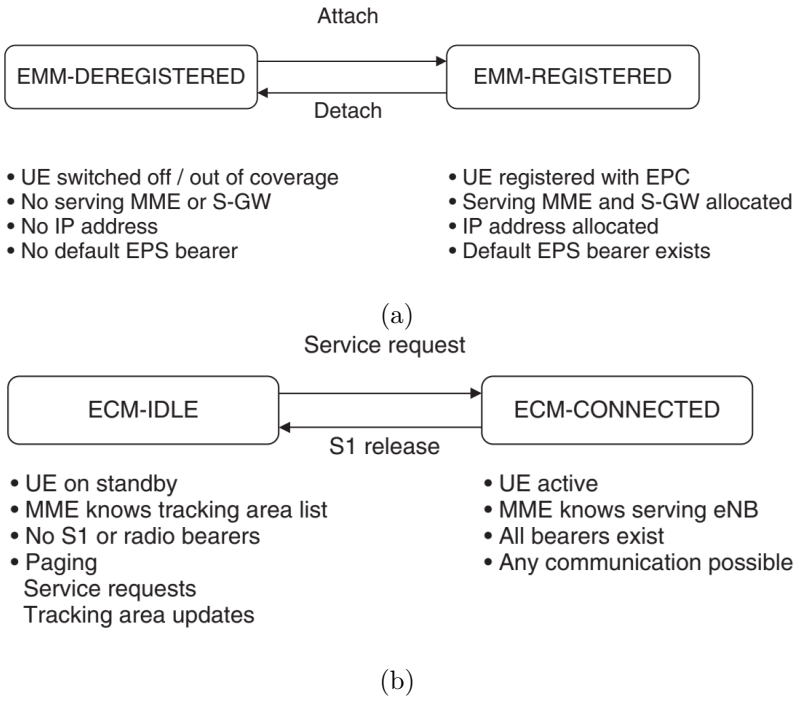


Figura 3.4: Diagramas de estado para EMM (a) y ECM (b) [30].

(RA-RNTI). En su solicitud, el UE encapsula el mensaje de tipo NAS Attach Request para obtener comunicación con el EPC (como muestra la figura 3.5). De esta forma se identifica con la red núcleo con el fin de acceder a los recursos de radio correspondientes. Para esto, si ya se había autenticado anteriormente, puede utilizar el último GUTI. Solo en caso de que no se cuente con un GUTI válido, el UE debe enviar su IMSI ya que es un identificador único que puede ser utilizado para realizar ataques dirigidos al suscriptor.

Una vez identificado en el MME, este determina si existe un contexto válido del usuario previamente establecido, o si es posible recuperarlo de otra entidad MME. De no encontrarse una asociación de seguridad válida, el proceso de registro continúa con el mecanismo Evolved Packet System-Authentication and Key Agreement (EPS-AKA). Este procedimiento en particular permite la autenticación mutua entre el usuario y la red LTE así como el establecimiento de una clave maestra a partir de la cual se derivan las claves para los servicios de confidencialidad e integridad.

Uno de los esquemas de cifrado más comunes para la autenticación en el EPS-AKA es el Milenage. Este utiliza una función núcleo de cifrado de bloque en el que tanto el tamaño del bloque como el tamaño de la clave son 128 bits. La misma está generalmente basada en la forma básica del algoritmo Advanced Encryption Standard (AES)[32].

Luego de la autenticación exitosa del usuario se lleva a cabo la actualización de los registros de localización en el HSS. Además, este último provee datos sobre la suscrip-

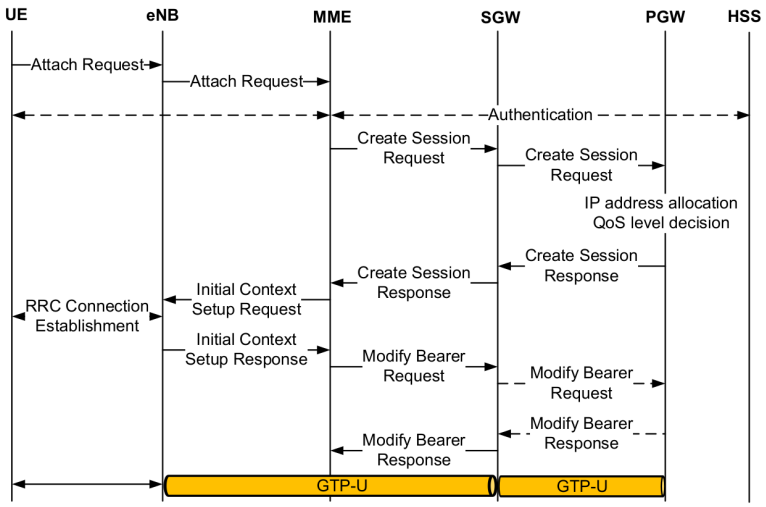


Figura 3.5: Proceso de registro (attach) [23].

ción respecto a las posibles conexiones PDN autorizadas y el perfil de QoS adquirido, necesarios para la creación del contexto de usuario en el MME.

Seguidamente, el MME realiza la selección y coordinación con el SGW y PGW para el establecimiento de la conexión PDN con la activación de, al menos, el servicio portador EPS por defecto. Si la dirección IP no es especificada por la suscripción, el PGW será el encargado de realizar su asignación de forma dinámica. Finalizadas las gestiones de señalización para la conexión PDN, quedan operativos los contextos de datos asociados al usuario en las tres entidades de la red troncal (MME, SGW y PGW).

Posteriormente, se realiza la creación del contexto en la E-UTRAN a través del mensaje Initial Context Setup Request que envía el MME al eNB, a través de la interfaz S1-MME. Y se envía igualmente el mensaje Attach Accept al UE conteniendo, por ejemplo, la dirección IP y el nuevo GUTI asignado y la lista de áreas de seguimiento donde es válido el registro recién completado. A su vez, el eNB prosigue con el envío del mensaje RRC Connection Reconfiguration para establecer el o los servicios portadores de radio correspondientes. Finalmente el UE confirma la terminación del proceso de registro enviando el mensaje de tipo NAS Attach Complete que llega al MME mediante los protocolos RRC y S1-AP.

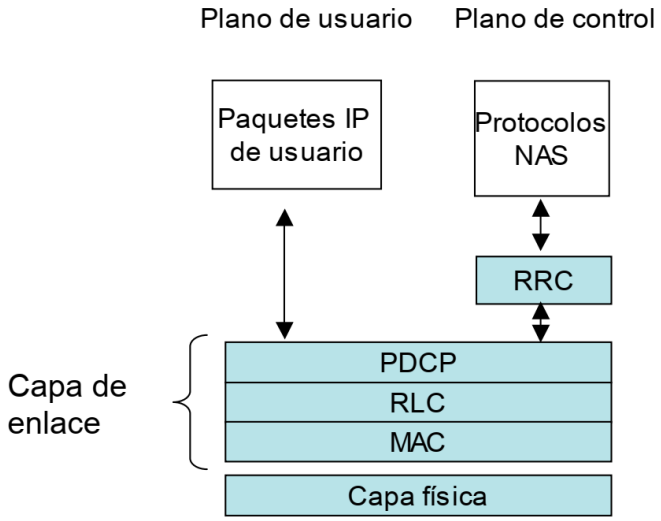


Figura 3.6: Protocolos de la interfaz radio de E-UTRAN [12].

3.5. Protocolos

En este apartado se da un resumen de las principales pilas de protocolos utilizadas en las interfaces que comunican los nodos descritos en el apartado anterior. Las pilas de protocolos se dividen en plano de usuario y plano de control para separar las funciones que dan soporte al envío de datos de usuario o paquetes IP y las que soportan las operaciones y procedimientos de cada interfaz, respectivamente.

3.5.1. Protocolos en la interfaz radio

La pila de protocolos de la interfaz de radio que conecta el eNB con el UE, como muestra la figura 3.6, representa las capas más inferiores del protocolo TCP/IP: capa física y capa de enlace.

La capa de enlace de datos engloba los protocolos PDCP, RLC y MAC tanto para el plano de usuario como para el plano de control. A continuación se describen brevemente las subcapas de la pila de protocolos con sus funciones específicas:

- Packet Data Convergence Protocol (PDCP): esta capa interactúa directamente con los paquetes IP y se encarga de la compresión de cabeceras, utilizando el algoritmo Robust Header Compression (ROHC), y el cifrado de la información de usuario para garantizar la integridad y confidencialidad de los datos. También identifica posibles paquetes duplicados y brinda al protocolo RRC el servicio de transferencia de mensajes de señalización denominado servicio portador de señalización o Signalling Radio Bearer (SRB).

- Radio Link Control (RLC): se encarga del control del enlace (capa 2) entre el UE y el eNB, para lo cual, puede usar uno de los siguientes modos:

  Transparent Mode (TM): es el modo más simple ya que no agrega encabezados de información referente a segmentación, secuencia, detección o corrección de errores. Es utilizado para procedimientos de voceo y transmisiones Broadcast.

  Unacknowledged Mode (UM): es utilizado para servicios dedicados que son sensibles al retardo, como VoIP y vídeo llamadas. Ofrece servicios de segmentación, concatenación y control de secuencia de paquetes pero no para corrección de errores.

  Acknowledged Mode (AM): ofrece fidelidad en la entrega al utilizar el mecanismo Automatic Repeat Request (ARQ) para la retransmisión de paquetes perdidos o dañados.

- Medium Access Control (MAC): se encarga de la administración de la interfaz de radioeléctrica, entre sus funciones principales está: mapeo entre los canales lógicos y los canales de transporte, multiplexación de servicios portadores en los canales de transporte, corrección de errores usando la técnica HARQ, asignación dinámica de recursos de red en base a parámetros de calidad y prioridades de servicio a través del algoritmos de calendarización (scheduling).

- Capa física: se especializa en la transmisión de los PDU de la capa MAC que recibe a través de los canales de transporte que proporciona. Entre sus funciones está: codificación de canal, modulación, procesamiento MIMO, sincronización en tiempo y frecuencia de la señal, mediciones de los parámetros del canal de radio, entre otras.

El plano de control, encargado de transportar los mensajes NAS entre el UE y la red núcleo, además de los protocolos mencionados anteriormente en este apartado, agrega un nivel de encapsulamiento con el protocolo RRC. Seguidamente se describen los elementos específicos de este plano en la interfaz de radio.

- Radio Resource Control (RRC) [33]: es el protocolo central para gestionar el uso de la interfaz de radio mediante intercambios del planos de control entre el eNodeB y los UE conectados. Entre sus principales funciones se tienen: gestión de los servicios portadores radio, soporte de funciones de movilidad (Selección/Reelección de celda y Handover), difusión (broadcast) de los bloques de información del sistema (SIB), reporte de mediciones y funciones de aviso a los terminales que no disponen de una conexión RRC establecida (Paging). También gestiona funciones de seguridad para el cifrado y la protección de la integridad del flujo de información entre el eNodeB y los UE.

- Señalización de los protocolos NAS [34]: Esta capa no pertenece estrictamente a la interfaz Uu, sin embargo hace uso de esta para transportar mensajes de señalización entre el MME y el equipo de usuario de forma transparente al estar encapsulado en la parte de datos de los mensajes RRC.

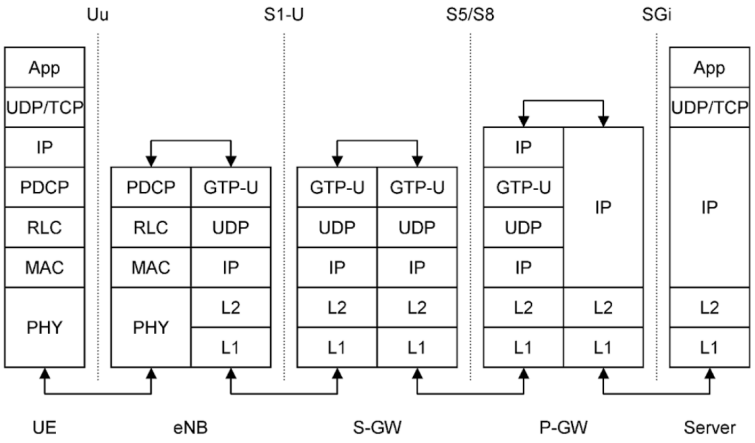


Figura 3.7: Pila de protocolos utilizada para intercambiar datos entre el servidor móvil y un servidor externo, cuando se utiliza una interfaz S5/S8 basada en GTP [35].

3.5.2. Otros protocolos en E-UTRAN y EPC

3.5.2.1. Plano de usuario

En la torre del plano de usuario de la E-UTRAN hay que destacar el protocolo GPRS Tunneling Protocol – User Plane (GTP-U). Este protocolo está encargado de encapsular los paquetes de datos de diferentes usuarios transportados por UDP (para evitar retrasar los datos) y multiplexarlos asignándoles un identificador llamado Tunnel End Point IDentifier (TEID), para poder ser localizados dentro del túnel que está asociado a un determinado EPS Bearer.

Constituye el protocolo de aplicación de la interfaz X2 y la interfaz S1-U que interconecta la E-UTRAN con el EPC. En el caso del EPC, todas las interfaces para el transporte de información de plano de usuario entre los diferentes elementos se soportan a través del protocolo GTP-U, excepto la variante de la interfaz S5/S8 basada en Proxy Mobile IPv6 (PMIPv6).

En la figura 3.7 se ejemplifica el uso de las pilas de protocolos del plano de usuario (basadas en GTP) en un intercambio de datos entre el terminal móvil y un servidor externo. El tráfico IP del plano de datos se encapsula en paquetes GTP en la capa GTP-U. Estos paquetes se canalizan a través del EPC hasta llegar al PGW donde se "extraen" del túnel y se pasan a algún nodo en Internet.

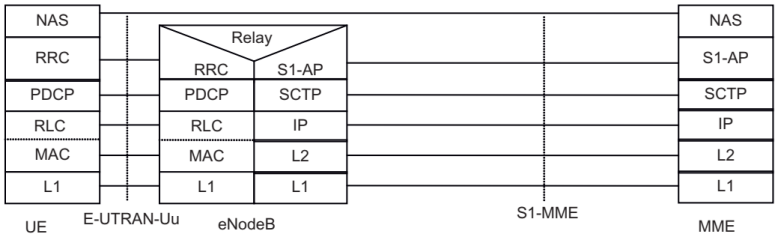


Figura 3.8: Pila de protocolos para plano de control en las interfaces E-UTRAN-Uu y S1-MME [38].

3.5.2.2. Plano de control

Respecto al plano de control, en figura 3.8 se puede apreciar la pila de protocolos que soporta los mensajes de señalización de tipo NAS a través de las interfaces Uu y S1-MME. Además de los ya tratados, se encuentran los protocolos S1 - Application Part (S1-APP) [36] y Stream Control Transmission Protocol (SCTP) [37] que interconectan el eNB con el MME.

El protocolo S1-AP tiene entre sus principales funciones:

- Gestión de servicios portadores.
- Control de movilidad y entre diferentes tecnologías de radio.
- Transporte del contexto de configuración entre el eNB y MME.
- Transferencia de información relacionada con las capacidades del UE, así como la ubicación(paging).

Para realizar el transmisión fiable de los mensajes, el protocolo S1-AP utiliza el servicio brindado por SCTP. Este protocolo de transporte está basado en TCP pero con adaptaciones que optimizan su función de entrega de mensajes de señalización de redes telefónicas sobre redes IP. Para esto incorpora, por ejemplo, soporte para multi-homing y multi-streaming permitiendo múltiples orígenes con diferentes direcciones IP y múltiples flujos paralelos de datos para una misma asociación, respectivamente [12].

Los mensajes del MME para la configuración de los UE son enviados a través de la interfaz S1-AP hacia el eNB, donde son encapsulados dentro de mensajes RRC y enviados por la interfaz aire. Esto funciona de forma similar a la inversa por lo que se dice que el eNB hace función de relay.

Dentro de la red núcleo encontramos otro importante protocolo del plano de control que es el GPRS Tunnelling Protocol - Control Plane (GTP-C) [39], también denominado como GTPv2-C. Sus funciones están enfocadas en gestionar las conexiones del plano de usuario en el EPC, esto incluye el control y administración de:

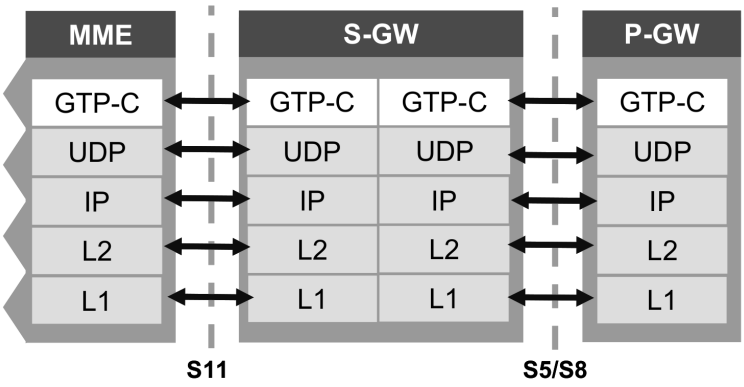


Figura 3.9: Pila de protocolos que sostiene a GTP-C en el plano de control del EPC [30].

- Parámetros de QoS.
- Túneles GTP-U (si se usa GTP en las interfaces S5/S8), conexiones PDN y servicios portadores EPS.
- Procedimientos de gestión de movilidad como transferencia de contextos de información de los usuarios.

Como se puede observar en la figura 3.8, el protocolo GTP-C puede ser usado en las interfaces S11 y S5/S8 de la arquitectura básica de LTE. Al igual que su homólogo en el plano de usuario, utiliza como protocolo de transporte UDP en lugar de TCP para evitar retrasos. Dado que las capas superiores se encargan de proporcionar confiabilidad con recuperación de errores y retransmisión. Por su parte, en las capas inferiores los paquetes IP pueden transportarse sobre una variedad de tecnologías L2 y L1, por ejemplo, Ethernet y ATM [30].

3.5.3. Canales lógicos, de transporte y físicos

Entre las capas que componen la pila de protocolos definida anteriormente para la interfaz aire del sistema LTE, existen interacciones que se estructuran a través de los llamados canales lógicos, de transporte y físicos. Las asignaciones admitidas entre los tipos de canales lógicos y los de transporte para el enlace descendente y el enlace ascendente, así como los canales físicos correspondientes, se muestran en la figura 3.10. Los tipos de canales, comenzando descendentemente, se describen brevemente a continuación:

- Canales lógicos: se establecen entre las capas RLC y MAC y define el tipo de información que se transporta, ya sea de usuario o señalización. Pudiendo ser un canal

de control o un canal de tráfico, si es usado para la transmisión de información de configuración y control o para los datos del usuario, respectivamente.

- Canales de transporte: la capa MAC utiliza servicios de la capa física en forma de canales de transporte. Diferentes canales lógicos pueden estar multiplexados en un canal de transporte y estos básicamente definen cómo y con qué características se transmite la información a través de la interfaz de radio. Los datos se organizan en bloques de transporte de tamaño dinámico que son transmitidos, generalmente y dependiendo de la existencia o no de multiplexación espacial (MIMO), en cada intervalo de tiempo de transmisión (TTI).
- Canales físicos: por último los canales físicos se encargan de los mecanismos para la transmisión y recepción a través del medio físico, o sea el enlace de radio. Los canales físicos se pueden clasificar en canales de tráfico y canales de control.

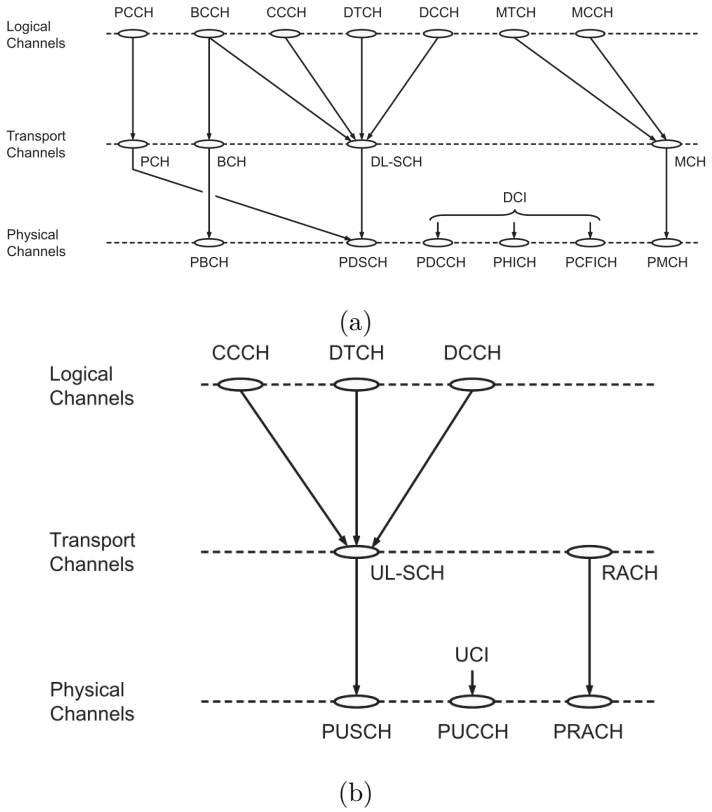


Figura 3.10: Mapa de canales para DL (a) y UL(b) [8].

Para soportar el manejo de prioridad, múltiples canales lógicos pueden ser multiplexados en un canal de transporte por la capa MAC, donde cada canal lógico tiene su propia entidad RLC. En el receptor, la capa MAC realiza la demultiplexación correspondiente y envía las PDU RLC a su entidad respectiva para la entrega en secuencia,

entre otras funciones.

3.6. Capa física

Las propiedades de la capa física, al determinar la eficiencia en el uso de los recursos radio, resuelve en gran medida las características de un sistema celular con respecto a la tasa de datos máxima, la latencia y la cobertura. En LTE, los fundamentos establecidos en esta capa constituyen una de las principales diferencias en relación a los sistemas celulares predecesores.

3.6.1. Esquemas de transmisión

La interfaz física de LTE se basa en OFDM para el DL y SC-FDMA para el UL. Estas técnicas, además de robustas, son adecuadas para entornos de ancho de banda flexible. Permitiendo realizar la adaptación del sistema para funcionar en diferentes bandas de frecuencias y anchos de banda disponibles.

Los modos de transmisión pueden ser FDD (bandas pareadas) o TDD (bandas no pareadas). Para FDD se definen diferentes rangos de frecuencias para DL y UL, mientras que en TDD todas las transmisiones están sobre el mismo rango de frecuencias pero separados en el tiempo. Cada método tiene sus ventajas y desventajas, por ejemplo, TDD permite economizar el espectro disponible al usar bandas de frecuencias no emparejadas, sin embargo tiene algo más de sobrecarga y latencia debido al cambio frecuente en el tiempo.

3.6.1.1. OFDM

El concepto básico de OFDM consistente en subdividir el espectro total de canales disponibles en un número de canales de 15 kHz, cada uno de los cuales lleva una subportadora. La capacidad disponible (el uso de estas subportadoras) se puede controlar en los dominios de tiempo y frecuencia al mismo tiempo. Sobre un conjunto de subportadoras se multiplexa un conjunto de símbolos.

La transmisión simultánea de todos los símbolos manteniendo la capacidad de separación de estos en recepción, se hace posible gracias a las propiedades de ortogonalidad de las subportadoras. Cada subportadora tendrá una velocidad de símbolo lenta y, en consecuencia, un tiempo de símbolo largo. Esto significa que se reduce la interferencia

entre símbolos. También se emplea la transmisión del denominado prefijo cíclico [1].

Estas características permiten que muestre un buen rendimiento en entornos de radio con afectaciones por reflejos mutitrayectorias. Otra de sus ventajas conociste en la sencillez y baja complejidad del diseño para el receptor. Además, dado que las subportadoras pueden asignarse a diferentes usuarios, es posible multiplexar datos de varios usuarios en un mismo intervalo de transmisión [40].

3.6.1.2. SC-FDMA

Por su parte, el esquema de transmisión para el enlace ascendente, SC-FDMA, funciona de manera similar a la del enlace descendente (OFDM), sin embargo tiene un Peak to Average Power Ratio (PAPR) más bajo que OFDM. Precisamente esta es una de las principales razones para elegir esta técnica para el UL ya que el amplificador de potencia en el UE puede fabricarse a un costo menor y permite que este realice un uso más eficiente de la energía.

SC-FDMA permite la separación ortogonal de las transmisiones del enlace ascendente también en el dominio de frecuencia. De esta forma, varios terminales pueden transmitir de forma simultánea sobre el intervalo del espectro que le fue asignado, posibilitando el acceso multiusuario tanto por división de tiempo (TDMA) como por división de frecuencia (FDMA). La separación ortogonal es beneficiosa, ya que evita en muchos casos la interferencia intracelular entre terminales.

3.6.2. Recursos de radio

Los recursos de radio en LTE están direccionados sobre una cuadrícula de tiempo-frecuencia de dos dimensiones para la transmisión de datos, denominada Resource Grid. Dentro de esta se ubican los bloques de recursos físicos o Physical Resource Blocks (PRB) cuya duración se corresponde con la de un slot de tiempo (0.5 ms). En el orden de la frecuencia abarca 12 subportadoras de 15 kHz cada una. A su vez, dentro de los bloques de recursos se ubican los elementos de recurso o Resource Element (RE), como se muestra en la figura 3.11, que constituyen la unidad básica ya que delimita la transmisión de un símbolo sobre una portadora.

La trama LTE, en el dominio del tiempo, tiene una duración de 10 ms. Cada trama está formada por 10 subtramas de 1ms y cada subtrama por dos intervalos de 0.5 ms

[1] Consiste en alargar la transmisión de cada símbolo OFDM a base de repetir, al principio del símbolo, una parte de la señal que se envía al final del mismo

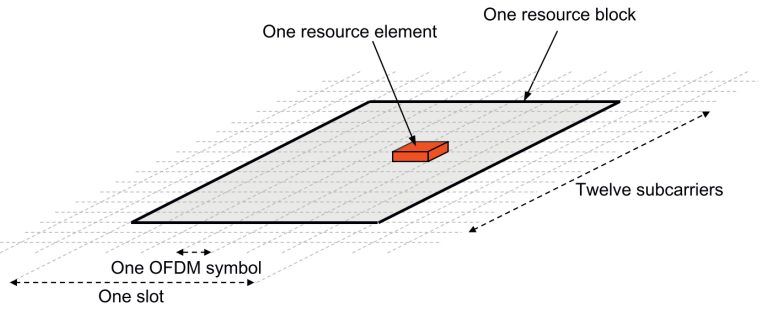


Figura 3.11: Bloque de recurso físico de tiempo-frecuencia LTE [8].

(tiempo de slot). Cuando se utiliza un prefijo cíclico (CP) normal cada bloque de recursos contiene 7 símbolos. En caso de ser extendido, se define una cantidad de 6 símbolos ya que aumenta la duración del CP para disminuir la interferencia intersimbólica en entornos de hostilidad para la señal.

La asignación dinámica de recursos permite que se entregue un número variable de RB a un determinado usuarios en diferentes momentos. Además, para optimizar la entrega según las condiciones del canal, se puede seleccionar diferentes esquemas de modulación y codificación, lo que resulta en tasas de datos variables para los usuario.

3.6.3. Anchos de banda y subportadoras

Una celda puede ser configurada con diferentes anchos de banda como muestra la tabla 3.2 . A mayor ancho de banda corresponde un mayor número de portadoras y por tanto puede ser enviada mayor cantidad de información. De esta forma, con 20 MHz, se logra que la estación base LTE opere a la velocidad de datos más alta posible. Cabe destacar que no todas las portadoras son ocupadas para la transmisión de información debido a que LTE utiliza bandas de guardia para disminuir la interferencia provocada por canales adyacentes.

3.6.4. Bandas de frecuencia E-UTRA

El estándar LTE ofrece cierta libertad en cuanto al uso de bandas de frecuencia identificadas para los modos de transmisión por FDD o TDD. En este sentido pueden ser usadas las definidas para un Release en específico, como las que se muestran en la tabla 3.3 para el Release 10, o incluso las definidas posteriormente o aún por definir. Siempre que los rangos cumplan con los requerimientos y restricciones de la versión que va a ser desplegada.

Bandas E-UTRA	Uplink (UL)	Downlink (DL)	Modo de duplexado
1	1920 MHz – 1980 MHz	2110 MHz – 2170 MHz	FDD
2	1850 MHz – 1910 MHz	1930 MHz – 1990 MHz	FDD
3	1710 MHz – 1785 MHz	1805 MHz – 1880 MHz	FDD
4	1710 MHz – 1755 MHz	2110 MHz – 2155 MHz	FDD
5	824 MHz – 849 MHz	869 MHz – 894MHz	FDD
6 [1]	830 MHz – 840 MHz	875 MHz – 885 MHz	FDD
7	2500 MHz – 2570 MHz	2620 MHz – 2690 MHz	FDD
8	880 MHz – 915 MHz	925 MHz – 960 MHz	FDD
9	1749.9MHz – 1784.9MHz	1844.9MHz – 1879.9MHz	FDD
10	1710MHz – 1770MHz	2110MHz – 2170MHz	FDD
11	1427.9 MHz – 1447.9 MHz	1475.9 MHz – 1495.9 MHz	FDD
12	699 MHz – 716 MHz	729 MHz – 746 MHz	FDD
13	777 MHz – 787 MHz	746 MHz – 756 MHz	FDD
14	788 MHz – 798 MHz	758 MHz – 768 MHz	FDD
15	Reservada	Reservada	FDD
16	Reservada	Reservada	FDD
17	704 MHz – 716 MHz	734 MHz – 746 MHz	FDD
18	815 MHz – 830 MHz	860 MHz – 875 MHz	FDD
19	830 MHz – 845 MHz	875 MHz – 890 MHz	FDD
20	832 MHz – 862 MHz	791 MHz – 821 MHz	FDD
21	1447.9 MHz – 1462.9 MHz	1495.9 MHz – 1510.9 MHz	FDD
22	3410 MHz – 3490 MHz	3510 MHz – 3590 MHz	FDD
23	2000 MHz – 2020 MHz	2180 MHz – 2200 MHz	FDD
24	1626.5 MHz – 1660.5 MHz	1525 MHz – 1559 MHz	FDD
25	1850 MHz – 1915 MHz	1930 MHz – 1995 MHz	FDD
...			
33	1900 MHz – 1920 MHz	1900 MHz – 1920 MHz	TDD
34	2010 MHz – 2025 MHz	2010 MHz – 2025 MHz	TDD
35	1850 MHz – 1910 MHz	1850 MHz – 1910 MHz	TDD
36	1930 MHz – 1990 MHz	1930 MHz – 1990 MHz	TDD
37	1910 MHz – 1930 MHz	1910 MHz – 1930 MHz	TDD
38	2570 MHz – 2620 MHz	2570 MHz – 2620 MHz	TDD
39	1880 MHz – 1920 MHz	1880 MHz – 1920 MHz	TDD
40	2300 MHz – 2400 MHz	2300 MHz – 2400 MHz	TDD
41	2496 MHz - 2690 MHz	2496 MHz - 2690 MHz	TDD
42	3400 MHz – 3600 MHz	3400 MHz – 3600 MHz	TDD
43	3600 MHz – 3800 MHz	3600 MHz – 3800 MHz	TDD

Tabla 3.3: Bandas operativas E-UTRA [41].

Ancho de banda total	Cantidad de PRB	Número de subportadoras	Ancho de banda ocupado
1.4 MHz	6	72	1.08MHz
3MHz	15	180	2.7 MHz
5MHz	25	300	4.5 MHz
10MHz	50	600	9 MHz
15MHz	75	900	13.5 MHz
20MHz	100	1200	18 MHz

Tabla 3.2: Anchos de banda de celdas compatibles con LTE.

[1]La banda 6 no es aplicable.

3.6.5. Indicadores para la caracterización de la señal

3.7. Indicador de calidad de la señal

El valor Channel Quality Indicator (CQI) es enviado por el terminal móvil hacia el eNB para informar la calidad con la que es capaz de percibir el canal de downlink. Está relacionado directamente con la tasa máxima de datos que el UE puede manejar logrando una relación de error de bloque (BLER) igual o menor al 10 %. En este sentido, como muestra la tabla 3.4, puede ser mapeado en términos del esquema de modulación y la tasa de codificación que un móvil puede recibir con éxito.

Como es lógico, para lograr aumentar la tasa de recepción exitosa y por ende el CQI, deben tenerse altos valores de la relación señal a ruido más interferencia (Signal to Interference plus Noise Ratio (SINR)). Sin embargo, otros factores como la categoría del UE (tecnología del receptor) pueden influir en la desempeño del celular [35].

El CQI puede ser referente a todo el ancho de banda del sistema o a sub-bandas, el tamaño depende, por ejemplo, del modo de operación utilizado (periódico o aperiódico) y del número de recursos físicos (N_{PRB}) ubicados en la banda disponible. La estación base utiliza el CQI recibido para la determinación del esquema de modulación y codificación y en apoyo de la programación dependiente de la frecuencia.

CQI	Esquema de modulación	Tasa de codificación (1/1024)	Bits de información por símbolo
0	n/a	0	0.00
1	QPSK	78	0.15
2	QPSK	120	0.23
3	QPSK	193	0.38
4	QPSK	308	0.60
5	QPSK	449	0.88
6	QPSK	602	1.18
7	16-QAM	378	1.48
8	16-QAM	490	1.91
9	16-QAM	616	2.41
10	64-QAM	466	2.73
11	64-QAM	567	3.32
12	64-QAM	666	3.90
13	64-QAM	772	4.52
14	64-QAM	873	5.12
15	64-QAM	948	5.55

Tabla 3.4: Interpretación de CQI [42].

3.8. Modulación y codificación adaptativas

En el estándar LTE, el eNB es capaz de seleccionar, con cierta inteligencia, el esquema de modulación y codificación (MCS) teniendo en cuenta las condiciones del canal que son reportadas por el UE mediante el CQI, para el caso del DL, y mediante sondeos de canal para el UL [43]. El objetivo es mejorar la throughput de datos entre el eNB y UE. Para hacer dicha adaptación también se tiene en cuenta la calidad de servicio requerida y el rendimiento de la celda.

Las tablas 3.5 y 3.6 fueron elaboradas según información definida por el 3GPP en

Índice MCS (I_{MCS})	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Orden de modulación (Q_m)	2	2	2	2	2	2	2	2	2	2	4	4	4	4
Índice de TBS (I_{TBS})	0	1	2	3	4	5	6	7	8	9	9	10	11	12

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4	4	4	6	6	6	6	6	6	6	6	6	6	6	6	2	4	6
13	14	15	15	16	17	18	19	20	21	22	23	24	25	26	Reservado		

Tabla 3.5: Tabla de modulación e índices TBS para PDSCH.

Índice MCS (I_{MCS})	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Orden de modulación (Q_m)	2	2	2	2	2	2	2	2	2	2	2	4	4	4
Índice de TBS (I_{TBS})	0	1	2	3	4	5	6	7	8	9	10	10	11	12

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4	4	4	4	4	4	4	6	6	6	6	6	6	6	6	Reservado		
13	14	15	16	17	18	19	19	20	21	22	23	24	25	26			

Tabla 3.6: Tabla de modulación e índices TBS para PUSCH.

[42]. Muestra el mapeo entre los índices MCS con la modulación y el índice del tamaño de bloque de transporte (TBS) correspondientes a DL y UL, respectivamente. Como es posible observar, el MCS puede tomar 32 posibles valores (desde 0 a 31). Un mayor valor se asigna cuando mejoran las condiciones del canal y puede indicar una modulación de orden superior, menor sobrecarga de codificación y bloques de transporte más grandes, por lo que mejora el throughput. Dado que los valores del 29 al 31 son reservados para uso futuro tanto para DL como UL, el valor 28 ofrece la mayor tasa de transmisión de datos.

3.9. Modos de transmisión multiantena

En el enlace descendente, la transmisión con múltiples antenas puede manejarse de diferentes formas o modos de transmisión. Esto, mostrados en la tabla 3.7, definen el tipo de procesamiento de múltiples antena que la estación base usará para sus transmisiones del PDSCH y por tanto el tipo de reconocimiento que deberá emplear el terminal móvil en la recepción.

Modo	Release LTE	Propósito
1	R8	SISO, transmisión por una única antena
2	R8	Diversidad de transmisión en bucle abierto [1]
3	R8	Multiplexación espacial en bucle abierto
4	R8	Multiplexación espacial en bucle cerrado [1]
5	R8	MIMO de múltiples usuarios
6	R8	Diversidad de transmisión en bucle cerrado
7	R8	Beamforming
8	R9	Beamforming de doble capa
9	R10	Multiplexación espacial de ocho capas

Tabla 3.7: Modos de transmisión para el enlace descendente.

[1]El término bucle cerrado o bucle abierto se refiere a si se forma o no un ciclo de retroalimentación del indicador de matriz de precodificación (PMI). Este indicador debe ser calculando y enviando por el UE al eNB a través de los reportes de información de estado del canal (CSI).

Por ejemplo, en el modo de multiplexación espacial el eNB usa múltiples antenas en la Tx y el UE debe utilizar la misma cantidad de antenas (categoría de UE 3 o superior) para realizar el demultiplexado. En una configuración base de LTE, el UE utiliza una sola antena para la Tx del UL (para mantener una baja complejidad) y hasta cuatro en la Tx del DL [44].

El parámetro Rank Indicator (RI) es la recomendación del UE para el número de capas a utilizar en la multiplexación espacial. Este valor se utiliza solamente cuando el UE está funcionando en modo MIMO con multiplexación espacial (modos de transmisión 3 y 4). Puede tener valores 1 o 2 al usar MIMO 2x2 o de 1 a 4 al usar MIMO 4x4. Este siempre se emplea asociado a uno o más reportes de CQI, indicándose que el reporte está asumiendo el uso de un RI en particular.

3.10. Métricas para la calidad del canal

En esta sección se presentan algunos de los principales indicadores que permiten caracterizar la señal de un enlace físico LTE: Reference Signals Received Power (RSRP), Signal to Noise Ratio (SNR) y Block Error Rate (BLER). Así como la relación entre

estos y su repercusión en el throughput logrado en capas superiores.

La potencia de la señal referencia recibida (RSRP) es una métrica relacionada con la intensidad de la señal específica de la celda que se utiliza. Se emplea para decidir la celda a la que se va a conectar el UE o si es necesario cambiar a otra (handover). Se calcula como el promedio de la potencia de los RE que portan la señal de referencia en un ancho de banda determinado [45].

Además de la necesidad de una buena potencia de recepción, la calidad del canal está determinada por la cantidad de ruido e interferencia que pueda afectar la señal. En este sentido, para que el estándar pueda alcanzar sus tasas de Tx pico, esta debe ser extremadamente buena, reflejándose en un alto SNR. Así lo indica la fórmula de Shannon (3.2) que plantea que la tasa de bits teórica máxima de un sistema depende solamente del ancho de banda y el SNR.

$$Bit\_rate = Bandwidth * log_2(1 + SNR) \tag{3.1}$$

Por su parte la fórmula para obtener el SNR ([46]) es bastante simple y está dada por:

$$SNR[dB] = S/N \tag{3.2}$$

Donde: S es la energía total de la señal en una subframe en un solo puerto de antena y N es la energía de ruido en un ancho de banda correspondiente al ancho de banda de Tx durante la duración del subframe.

Luego, un mayor valor de SNR significa que la señal llega con mejor potencia y/o menor ruido, lo cual influye en el esquema de Tx a escoger. De esta forma, para poder emplear satisfactoriamente una modulación con un alto número de símbolos, se requiere de buenas condiciones de SNR para evitar la interferencia intersimbólica. La recepción de bits en error se representa mediante el indicador BLER que indica el porciento de error en los bloques de transporte recibidos. Como se ha tratado en secciones anteriores, el UE selecciona un CQI con relación a este parámetro y con el objetivo de que no supere la tasa del 10 %.

De forma general, a mayor valor de RSRP y SNR, se obtendrá un menor BLER y por tanto el UE podrá reportar un mejor CQI. Permitiendo de esta forma alcanzar mayores tasas de Tx de datos. En las especificaciones técnicas del 3GPP [46] y [47] se ofrecen valores de referencia de los parámetros vistos para el eNB y el UE, respectivamente.

Capítulo 4

Software Defined Radio (SDR)

La tecnología SDR ha tomado su posición en un mundo que promete sistemas de comunicación inalámbricos universales y programables, mejorando así las ideas de diseños tradicionales de radio usando hardware fijo y específico. Con esta, las comunicaciones inalámbricas han evolucionado al aprovechar flexibilidad, programabilidad y bajo costo del software en lo que algunas fuentes consideran la tercera generación [1] en las comunicaciones.

En el presente capítulo se ofrece una introducción y descripción de la plataforma SDR y tecnologías relacionadas como USRP, GNU Radio y UHD. Posteriormente se analiza la biblioteca srsLTE como plataforma SDR para el despliegue de una red LTE.

4.1. Introducción y breve historia de los SDR

Definida como una tecnología de radio en el que algunas o todas las funciones de la capa física están definidas por software, se encuentra en auge de desarrollo y continúa siendo tema de discusión común hasta el día de hoy. Normalmente, la plataforma SDR emplea un conjunto de dispositivos de hardware programables, como las matrices de puertas programables de campo (FPGA) y los procesadores de señal digital (DSP) [48].

Al separar la arquitectura y las funciones del hardware del sistema, SDR puede realizar múltiples funciones mediante un software basado en una plataforma de hardware relativamente universal. Puede programar la frecuencia de funcionamiento, el ancho de banda del sistema, la modulación y la codificación de fuente [49], así como obtener sistemas escalables y modulares al poder integrar módulos de software.

La atractiva idea de implementar varios modos de transmisión simplemente recon-

[1] Antecedida por las transformaciones fijo-a-móvil y analógica-a-digital.

figurando el radio con diferentes software fue, en un inicio, desarrollada principalmente en el área militar y es un concepto que ha estado madurando lentamente. J. Mitola definió por primera vez el área y también acuñó el término en 1991, mientras que Steinbrecher realizó investigaciones iniciales sobre la idoneidad de *Software Radio* para implementar estaciones base inteligentes, flexibles y de múltiples estándares [50].

Poco a poco, muchos mercados fueron interesándose y acogiendo a los SDR luego de la aparición de avances tecnológicos como los circuitos integrados RF (RFICs), impulsados por compañías como Analog Devices, y FPGA rentables con uso intensivo de DSP, de empresas como Xilinx.

Hoy en día, los SDR se han convertido en estándar de facto de la industria en mercados como inteligencia de señales, guerra electrónica, prueba y medición, comunicaciones de seguridad pública, monitoreo de espectro y comunicaciones militares [51]. La solución programable se ha extendido ampliamente ya que admite el rápido desarrollo de estándares inalámbricos y un corto tiempo de comercialización [52].

4.2. Arquitectura SDR

Los dispositivos SDR cuentan idealmente con múltiples bandas y modos de arquitecturas abierta con el fin de poder abarcar un mayor rango de funciones inalámbricas, que se activan simplemente al cargar el software correspondiente.

Una plataforma SDR básica puede incluir: una antena, un módulo de radiofrecuencia (RF) multibanda, un convertidor de señal analógica a digital (ADC) y viceversa (DAC) de banda ancha, procesadores DSP, entre otras expansiones; como muestra la figura 4.1. Esta tecnología mueve los convertidores ADC y DAC más cerca de la RF, desde la base a la frecuencia intermedia o incluso a la RF, y reemplaza los circuitos digitales dedicados por dispositivos programables DSP o FPGA [53].

Un ejemplo de sistema SDR puede construirse combinando un dispositivo de radiofrecuencia programable (por ejemplo, un USRP de Ettus Research [55]) con una plataforma de procesamiento de banda base a través de una interfaz adecuada. Como plataforma de procesamiento de banda base suelen utilizarse procesadores de propósito general (GPP) para completar funciones como transceptor inalámbrico, generación de señal, modulación/demodulación, codificación/decodificación, etc. [20].

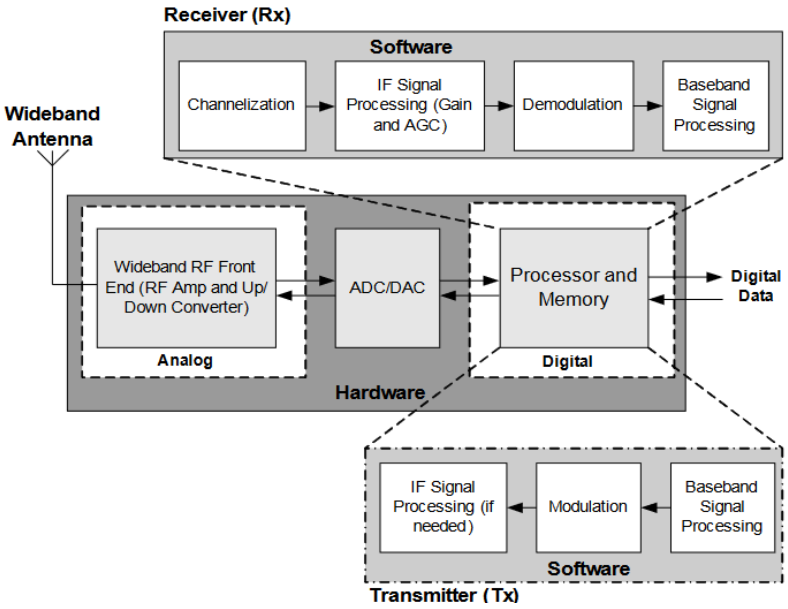


Figura 4.1: Arquitectura de hardware y software SDR [54].

4.3. Enfoque de CPU de propósito general (GPP)

Existen algunas clasificaciones para implementar SDR dependiendo de su arquitectura y el uso de GPP, DSP y/o FPGA para realizar el procesamiento de las señales [56]. Una de las más ampliamente utilizadas es la que emplea el enfoque GPP. Este, define el uso una computadora con un procesador convencional para realizar el tratamiento y obtención de la señal digital. Dejando básicamente al dispositivo RF conectado las funciones de conversión para su inmediata transmisión al aire.

Puede sonar contradictorio el hecho de usar un procesador externo cuando muchos de los dispositivos integran potentes DSP Y FPGA programables que pudieran cumplir las funciones de procesamiento de la señal. Sin embargo existen ventajas que han impulsado el uso de este enfoque [57] entre ellas se encuentran:

- Portabilidad: Escribir algoritmos para CPU de propósito general nos permite reutilizar el código en otras arquitecturas y sistemas operativos. Además, brinda mayor flexibilidad para la elección de hardware necesario para cumplir con cierta tarea al estar programada a alto nivel.
- Herramientas de desarrollo: La disponibilidad de herramientas de desarrollo para uso en computadoras convencionales se encuentra más disponible y bajo licencia de código abierto en muchos casos.
- Integración de aplicaciones. Se tiene más flexibilidad para comunicar e integrar

el software que procesa la señal RF con otras aplicaciones o servicios disponibles en la misma computadora.

Entre las desventajas podemos mencionar el alto consumo de energía para un objetivo de rendimiento. Además, dependiendo el CPU que se utilice, pueden obtenerse diferencias de latencias en el procesamiento. Por ejemplo, para realizar procesamientos complejos en tiempo real, es recomendable que el CPU sea compatible con técnicas multi-core y multi-threading.

Por su parte, los FPGA son útiles para realizar programas con procesamiento concurrente y tienen multiplicadores de hardware dedicados que disminuyen la latencia. Sin embargo, respecto al enfoque GPP, el desarrollo de software es mucho más complejo y tardado [58].

4.3.1. USRP

Los USRP son un representativo ejemplo del enfoque mencionado. Constituyen una gama de radios definidas por software diseñadas y vendidas por Ettus Research y su empresa matriz, National Instruments. Uno de los propósitos de esta familia de dispositivos de radio es ofrecer precios asequibles para potenciar su uso en universidades, laboratorios de investigaciones y aficionados.

Un USRP funciona como una interfaz de RF para una computadora que ejecuta software con funciones de procesamiento de señal (por ejemplo, GNU Radio, Matlab o LabView), convirtiendo las ondas de radio recogidas por una antena en copias digitales que el software puede manejar, o inversamente, convirtiendo una onda sintetizada por la computadora en una transmisión de radio.

El diseño básico de un USRP suele estar conformado por una motherboard que trae embebidos los ADCs, DACs y una FPGA de alto rendimiento. Los convertidores están conectados a unas interfaces de radio intercambiables o tarjetas hijas, como muestra la figura 4.2, mientras la FPGA se conecta mediante determinada interfaz al host de procesamiento.

La filosofía de diseño detrás del USRP ha sido realizar el procesamiento referente a la forma de onda, como modulación/demodulación, entre otros, en el CPU de la computadora. Mientras que las operaciones de propósito general que requieren altas velocidades como la conversión A/D (D/A), decimación [1] y la interpolación de la señal, se realizan en el FPGA [60].

Para el desarrollo del presente trabajo se dispone para su uso de los equipos USRP B210 y X310 (ver figura 4.3), sobre los cuales se aborda en próximas secciones.

[1] Proceso de reducir la frecuencia de muestreo de una señal.

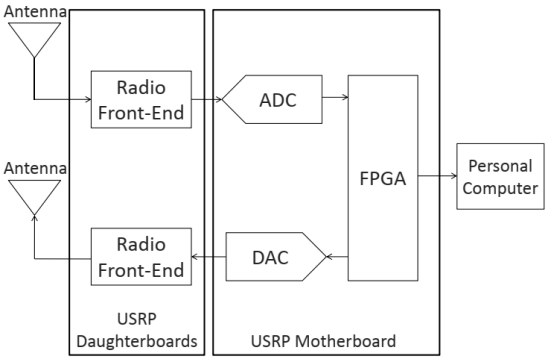
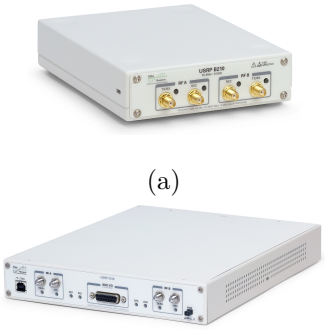


Figura 4.2: Arquitectura USRP simplificada [59].



(a)

(b)

Figura 4.3: Radios USRP B210 (a) y X310 (b).

4.3.2. USRP B210

El USRP B210 [61] ofrece una cobertura de frecuencia continua desde 70 MHz hasta los 6 GHz con un transceptor de conversión directa integrado que proporciona hasta 56MHz de ancho de banda en tiempo real. Un FPGA Xilinx Spartan6 abierto y reprogramable y una conveniente conectividad USB 3.0 SuperSpeed. Es completamente soportado su uso con GNU Radio o aplicaciones que usan la API UHD.

Entre sus especificaciones también se encuentran:

- Dos canales de recepción y dos de transmisión, Half o Full Duplex.
- Capacidad MIMO 2x2 totalmente coherente.
- Hasta 56 MHz de ancho de banda instantáneo en 1x1 y hasta 30.72 MHz con MIMO 2x2.

4.3.3. USRP X310

Por su parte, sobresaliente por ser escalable y de alto rendimiento, el USRP X310 [62] posee dos ranuras de placas hija con ancho de banda extendido que cobren hasta los 6 GHz con posibilidad de hasta 120 MHz de frecuencia de muestreo. Tiene varias opciones de interfaces de alta velocidad para la comunicación con el host de procesamiento y no se puede pasar por alto su FPGA Kintex-7. Es completamente compatible con los controladores UHD y por ello con una gran cantidad de plataformas y proyectos de código abierto. En la figura 4.4 se muestra un diagrama de la estructura interna de este radio.

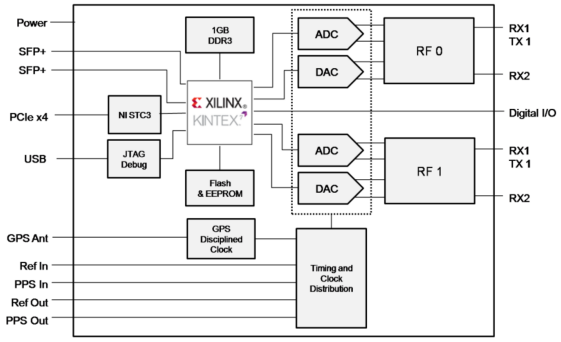


Figura 4.4: Diagrama de estructura interna del x310 [62].

Entre las prestaciones que ofrece también se encuentra:

- Múltiples interfaces de alta velocidad:

  Ethernet dual de 10 Gigabits: 2x RX (200 MSps por canal).

  Ethernet dual de 10 Gigabits: 4x RX (80 MSps por canal).

  PCIe Express: 200 MS/s Full Duplex.

  ExpressCard (computadora portátil): 50 MS/s Full Duplex.

  Dual 1 Gigabit Ethernet: 25 MS/s Full Duplex.

- Dos ranuras de placa hija RF de ancho de banda amplio:

  Ancho de banda de hasta 160MHz cada uno (tarjetas CBX, WBX, SBX).

  La selección de la placa hija cubre DC a 6 GHz.

- Arquitectura de reloj:

  Frecuencia de muestreo configurable.

  GPSDO[1] opcional.

[1]Es un oscilador de cristal disciplinado por GPS y controlado por horno que proporciona una referencia de 10 MHz de alta precisión y 1 señal PPS.

Operación coherente con OctoClock y OctoClock-G.

Las tarjetas hijas que tiene instalado el dispositivo X310 con el que se cuenta es una SBX-120 que tiene entre sus características:

- Rango de frecuencia: 400 MHz a 4.4 GHz.
- Rango de ganancias de transmisión: 0-31.5dB.
- Rango de ganancias de recepción:0-31.5dB.
- Ancho de banda: 120 MHz, para RX y TX.

4.4. GNU Radio

La plataforma de software GNU Radio [63], gratuita y de código abierto, es una de las más populares para diseñadores e investigadores en esta área de las comunicaciones inalámbricas. Está basada en GPP y es compatible con los dispositivos USRP.

Es un kit de herramientas de desarrollo de software gratuito que proporciona bloques de procesamiento de señales para implementar sistemas de radios definidos por software. Se puede usar con hardware de RF externo o sin este en un entorno similar a una simulación.

Con este es posible construir y ejecutar aplicaciones para el procesamiento de señales RF. Sus aplicaciones se reconocen generalmente como "diagramas de flujo" al formar una serie de bloques de procesamiento de señales conectados entre sí, que describen un flujo de datos. Al igual que con todos los sistemas de SDR, la reconfigurabilidad es una característica clave. En lugar de usar diferentes radios diseñados para propósitos específicos pero dispares, se puede usar uno único de propósito general como el front-end, y el software de procesamiento de la señal, maneja el procesamiento específico de la solicitud.

4.5. Interfaz USRP Hardware Driver (UHD)

UHD [6] es una API para desarrollo de aplicaciones SDR compatible con los equipos USRP. Al usarse una interfaz de software común, la comunicación con estos radios posibilita la reutilización e intercambio de código manteniendo cierta abstracción del equipo USRP con el que se cuente. Está disponible en código abierto para los sistemas operativos Linux, Windows y Mac OS y es compatible con varias plataformas como RFNoC, Radio GNU, LabVIEW y Matlab/Simulink.

Mediante comandos UHD es posible encontrar dispositivos en un sistema USRP e instanciarlos con parámetros deseados. Establece y obtiene propiedades de radio como la ganancia, amplitud, frecuencia central, frecuencia de muestreo y tiempo. Es capaz de transmitir/enviar muestras mediante operaciones de lectura/escritura del Sistema Operativo estándar. Creando una transmisión de envío o recepción entre la computadora y el FPGA en el USRP. También admite mensajes de control y administración (por ejemplo, desbordamiento (*overflow*), comando de error de subflujo (*underflow*) y errores de secuencia) [58].

4.6. Biblioteca srsLTE como plataforma SDR

En la sección 2.2.4 se ofrece una breve introducción sobre la herramienta de software srsLTE y sus potencialidades para el despliegue de una red LTE bajo un enfoque SDR. En este apartado se indaga un poco más en su estructura, funcionamiento y capacidades para el uso eficiente de los recursos computacionales.

El diseño de srsLTE, al ser modular, permite la intervención en partes del código sin que esto afecte a los demás módulos. Como muestra la figura 4.5 , los componentes están estructurados siguiendo cierta jerarquía para su organización.

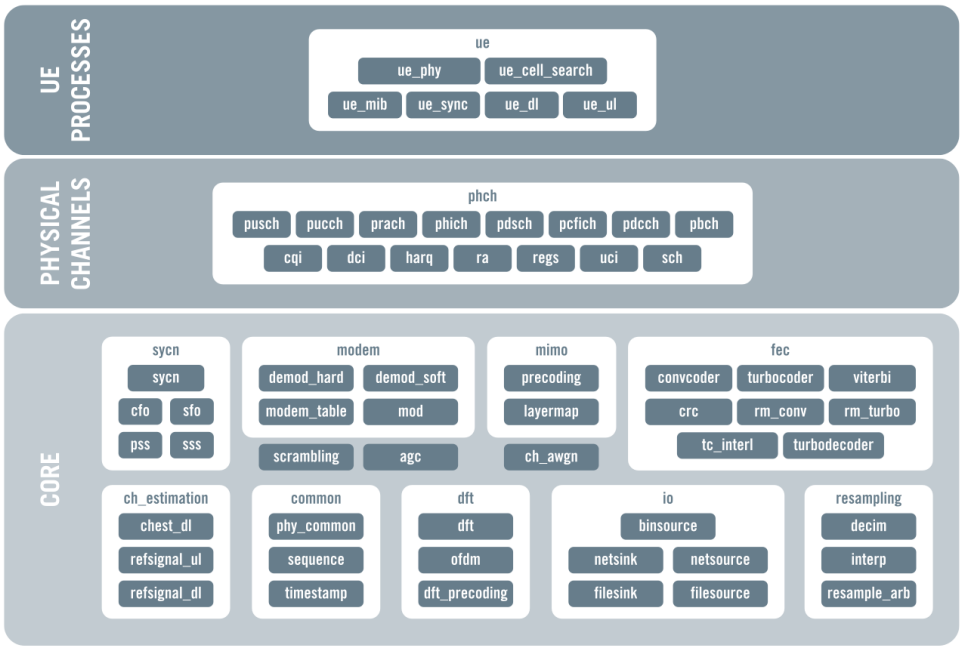


Figura 4.5: Diagrama de módulos de la biblioteca srsLTE [4].

En el módulo Core se encuentran los principales bloques de código para la implementación de las funciones de capa física como son la turbo codificación y decodificación convolucional, modulación/demodulación, sincronización, estimación de canal, generación de señales de referencia, procesamiento OFDM Y SC-FDMA, etc. Otro de los módulos agrupa los canales físicos de los enlaces ascendentes y descendentes, así como algunas clases de uso común. A su vez, las clases para el funcionamiento del módulo srsUE se organizan funcionalmente.

Al estar orientada principalmente a GPP, donde la memoria es barata, y siendo más eficiente el acceso a memoria que el cálculo computacional, se hace un uso extensivo de estructuras LUT (*look-up tables*) con una gran cantidad de señales pregeneradas. Por ejemplo las señales de referencia y algunas señales del PUCCH pueden ser generadas por cada subtrama y combinación de datos de entrada [18] .

Otro de los principios para el eficiente rendimiento computacional en esta biblioteca escrita en C es la promoción del uso de operaciones con procesamiento en paralelo del tipo *single instruction multiple data* (SIMD). Útiles principalmente en el receptor donde se ejecutan las partes más exigentes, por ejemplo, el estimador y ecualizador de canal, el demodulador y decodificador Viterbi y especialmente el turbo decodificador, cuya demanda de recursos es directamente proporcional al ancho de banda definido.

Capítulo 5

Desarrollo

En el presente capítulo se describen los detalles del despliegue de la herramienta srsLTE. Se detallan requerimientos de hardware y software, pasos para completar su instalación, parámetros de configuración establecidos, adaptaciones realizadas, así como su integración con los equipos USRP de Ettus Research, con los que se cuenta.

Además, diseñados para permitir una caracterización del sistema obtenido, se exponen los escenarios de prueba a desplegar en el próximo capítulo. Describiéndose las herramientas y aplicaciones involucradas.

5.1. Ambiente de hardware

La plataforma srsLTE, como biblioteca SDR, utiliza el enfoque de CPU de propósito general. Por lo que para su despliegue se utilizan computadoras personales que realizan el procesamiento de la señal antes de ser enviada al aire utilizando un dispositivo RF, en este caso, un USRP. Para su descripción, la configuración de hardware se divide en dos elementos: nodo de procesamiento e interfaz de radio.

5.1.1. Nodo de procesamiento

Para host de procesamiento se cuenta con dos posibles configuraciones de hardware, las cuales se describen y denotan a continuación.

Máquina A: Dell Precision 3630 (desktop):

- Procesador: 8th generation Intel Core i7
- RAM: 32GB
- Tarjeta gráfica: NVIDIA Quadro P5000

- Conexión a USRP x310: tarjeta de red PCIe de 10 Gigabit

Máquina B: Dell Latitude 7490 (laptop):

- Procesador: 7th generation Intel Core i7
- Tarjeta gráfica: Intel® UHD Graphics 620
- RAM: 8GB
- USB Type C™ con Thunderbolt™ 3
- Conexión a USRP B210: USB 3.1 de 1.ª gen. (uno con PowerShare)

5.1.2. Interfaz de radio

La relación de los equipos USRP que se van a usar para desplegar los módulos de la plataforma srsLTE se ofrece a continuación:

- srsENB y srsEPC: USRP X310 con tarjeta hija SBX-120 (400 MHz - 4.4 GHz, 120 MHz BW)
- srsUE: USRP B210

5.1.2.1. Oscilador de referencia y fuente de reloj

Los equipos USRP pueden funcionar con una señal de referencia externa o interna, indicación que se envía a través del driver UHD. En este caso, se utiliza una señal de 10 MHz de alta precisión y una señal de 1 PPS (*Pulse Per Second*) ofrecida por el módulo GPS Disciplined Oscillator (GPSDO) (*PCB-Mounted GPS-Disciplined OCXO*) [1] con temperatura controlada montado en la placa de los dispositivos. Este módulo se integra con una antena GPS activa de 5 voltios [2] proporcionando un aumento en la precisión de la frecuencia y alineación de temporización global.

Ambos equipos de radio se conectan a una fuente de electricidad con sus cables de alimentación pues, aunque el B210 puede funcionar solamente usando la conexión USB al host, requiere de este paso para poder usar el módulo GPSDO especificado.

Para comprobar el estado de los dispositivos, se ejecutaron las pruebas de verificación de funcionamiento recomendadas por el fabricante como punto de referencia [3]. Estos completan las pruebas de forma satisfactoria.

[1] Ver https://www.ettus.com/all-products/gpsdo-mini/.

[2] Ver https://www.ettus.com/all-products/gps-ant-5v/.

[3] Ver https://kb.ettus.com/Verifying_the_Operation_of_the_USRP_Using_UHD_and_GNU_Radio

5.1.2.2. Antenas

Figura 5.1: VERT2450 dual-banda: 2.4 GHz a 2.48 GHz y 4.9 GHz a 5.9 GHz.

Para la transmisión por aire se utilizan antenas VERT2450 (ver figura 5.1). Al utilizarlas durante la Tx y RX se orientan formando un ángulo de 90 grados para reducir la interferencia entre ambas. Los equipos SDR con los que se cuenta son USRP X310 y B210, cuyas especificaciones técnicas se ofrecieron en el capítulo anterior (ver sección 4.3.1).

5.1.3. Equipo UE comercial

Para las pruebas de desempeño de la plataformas se emplea también un UE comercial. En este caso se cuenta con el Samsung Galaxy S6 [1], un teléfono de buen desempeño con CPU Octa-Core a 2.1 GHz y 3 GB de RAM. Posee una amplia compatibilidad con varias bandas LTE FDD: B1, B2, B3, B4, B5, B7, B8, B12, B17, B18, B19, B26 y B28.

5.2. Ambiente de software

La biblioteca srsLTE permite construir una red móvil LTE de extremo a extremo de forma gratuita y con código abierto. Consta de tres módulos principales que pueden ser ejecutados sobre el sistema operativo Linux y equipos SDR: srsUE, srsENB y srsEPC.

En el momento del desarrollo de este trabajo, se encontraba como última versión disponible de la suite la 19.06, siendo esta la utilizada. Para realizar el despliegue se acondicionan las computadoras descritas en la sección anterior con el SO Ubuntu en su versión estable 18.04 LTS (64 bits). Posteriormente, se instalan algunas bibliotecas requeridas para la compilación del proyecto utilizando el siguiente comando:

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

Se recomienda también la instalación de la herramienta *Git*[2] para el control de versiones.

[1] Ver https://www.samsung.com/mx/smartphones/galaxy-s6-g920i/.

[2] Disponible en https://git-scm.com/.

5.2.1. Driver UHD

El controlador UHD es necesario para la comunicación entre la PC y los equipos USRP de Ettus Research. En este caso se utilizará la versión UHD v3.10.3 recomendada por la comunidad de la herramienta. Para esto se obtiene, compila e instala el código fuente correspondiente usando los siguientes comandos.

```
$ git clone git://github.com/EttusResearch/uhd.git
$ git checkout ef15767
$ mkdir host/build && cd host/build && cmake ../
$ make all && sudo make install
```

5.2.2. Biblioteca srsGUI

Parte de la información que arroja srsLTE en su funcionamiento puede ser graficada usando la biblioteca srsGUI, igualmente gratuita y de código abierto. Utiliza QT y Qwt para proporcionar varias gráficas con información útil que se actualiza en tiempo real. Antes de compilar es necesario instalar algunos paquetes requeridos:

```
$ sudo apt-get install libboost-system-dev libboost-test-dev libboost-↩
thread-dev libqwt-dev libqt4-dev
```

Posteriormente se descarga e instala el código fuente:

```
$ git clone https://github.com/srsLTE/srsGUI.git
$ mkdir srsgui/build/ && cd srsgui/build && cmake ../
$ make all && sudo make install
```

Para ejecutar una serie de pruebas que verifican el correcto funcionamiento de la biblioteca, se ejecuta el comando `make test`.

5.2.3. srsLTE: obtención, compilación y configuración

Para obtener y compilar el código fuente alojado en la página oficial de la biblioteca srsLTE en Github [4]:

```
$ git clone https://github.com/srsLTE/srsLTE.git
$ mkdir srsLTE/build/ && cd srsLTE/build && cmake ../ && make all
```

Una vez compilado sin errores, el comando `make test` nos ayuda a verificar que la compilación se haya completado correctamente y de esta forma dar paso a la instalación:

```
$ sudo make install
$ srslte_install_configs.sh user
```

De esta forma queda instalada en el sistema la biblioteca srsLTE con sus aplicaciones srsUE, srsENB y srsEPC. Con la última línea, se copian los archivos de configuración predeterminados en el directorio oculto `~/.config/srslte` ubicado en la carpeta raíz del usuario desde el que se haya ejecutado el comando. Estos archivos pueden mantenerse en este directorio para que el sistema los cargue automáticamente o puede definirse su origen directamente en el comando de ejecución. En próximas secciones se expondrá las modificaciones realizadas a la configuración para obtener el comportamiento deseado.

5.2.3.1. Estación base: srsENB

La aplicación srsENB ofrece la implementación en software para desplegar un eNB de LTE haciendo uso de un hardware SDR para la transmisión de las señales de radio al aire. Esta nodo funcional utiliza las interfaces estándar S1AP y GTP-U para conectarse a una red núcleo, por ejemplo srsEPC, y crear una celda LTE local. A dicha celda pueden conectarse dispositivos de usuario que pueden ser, tanto teléfonos celulares comerciales compatibles con 4G, como un despliegue de la aplicación srsUE en otra máquina.

Esta implementación es compatible, de forma alineada, con el Release 10 de LTE. Entre sus prestaciones encontramos:

- Compatible con el modo FDD y los anchos de banda 1.4, 3, 5, 10, 15 y 20 MHz.
- Modos de transmisión compatibles: 1 (antena única), 2 (diversidad de transmisión), 3 (Diversidad de Retraso Cíclico, CCD) y 4 (multiplexación espacial en bucle cerrado).
- Soporte para servicio multimedia de difusión y multidifusión evolucionado (eMBMS).
- Captura de paquetes de capa MAC con Wirehark.
- Soporte de retroalimentación periódica y aperiódica de CQI.
- 150 Mbps de DL en MIMO TM3/TM4 de 20 MHz con UE comercial.
- 75 Mbps de DL usando SISO con celular comercial.
- 50 Mbps de UL en 20MHz con celular comercial.

Para configurar el eNB que se va a desplegar es necesario editar el archivo de configuración principal (`~/.config/srslte/enb.conf`). De esta forma se pueden controlar parámetros de la celda como son: frecuencias de funcionamiento, potencia de transmisión, identificadores de la celda, entre otros. En la misma ubicación existen otros archivos de configuración avanzada para el SIB (`sib.conf`), recursos de radio (`rr.conf`) y servicios portadores de datos (`drb.conf`).

En el fragmento de configuración 5.1, se muestran y describen los parámetros principales establecidos para obtener el comportamiento deseado, los no mostrados mantienen su valor por defecto.

```
[enb]
mcc = 901
mnc = 55
mme_addr = 127.0.1.100
n_prb = 100
tm = 1
nof_ports = 1
[rf]
dl_earfcn = 3400
# dl_freq #ul_freq
tx_gain = 15
# rx_gain = AGC
device_name = UHD
device_args = auto
[pcap]
enable = true
filename = /tmp/enb.pcap
[gui]
enable = true
[scheduler]
#pdsch_mcs  #pusch_mcs
[expert]
metrics_csv_enable   = true
metrics_csv_filename = /tmp/↩
enb_metrics.csv
```

Fragmento 5.1: Configuración de srsENB.

Primeramente, y debiendo coincidir con los seleccionados para personalizar la tarjeta USIM (ver sección 5.2.4), se establecen los códigos de la red LTE: MCC y MNC. En caso de que el MME se encuentre en una máquina diferente al eNB, debe ser indicada su dirección IP usando `mme_addr`. Durante las pruebas a realizar, algunos de los parámetros se irán variando para el adecuado estudio del sistema, entre estos están: el número de PRB a emplear (`n_prb`), ganancias en la Tx (`tx_gain`) y Rx (`rx_gain`), valores fijos forzados para MCS del PDSCH (`pdsch_mcs`) y PUSCH (`pusch_mcs`).

La frecuencia de transmisión de la interfaz de radio puede ser definida mediante un código EARFCN seleccionado en `dl_earfcn` o bien definiendo la frecuencia de DL (`dl_freq`) y/o UL (`ul_freq`). En este trabajo se empleará el código EARFCN predeterminado 3400, el cual se corresponde con la banda 7. Utiliza como frecuencias centrales (modo TDD) 2685 MHz para DL y 2565 MHz para UL.

Para la supervisión del sistema durante su ejecución, se habilita la interfaz visual ofrecida por srsGUI. Se activa de igual forma la captura de paquetes de capa MAC mediante archivos `.pcap` que pueden ser visualizados usando la herramienta Wireshark. Para un futuro análisis se activa el registro de logs y de las métricas arrojadas por el eNB: *cqi, mcs, brate, bler y snr* (ver sección 3.10).

Respecto al SO donde se ejecuta el srsENB, es recomendable deshabilitar la opción de escalado de frecuencias de los núcleos del CPU para mejorar el rendimiento. Para esto se usa un script en Bash con el siguiente código:

```
#! /bin/bash
 for file in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do ↩
    echo "performance" > $file; done
```

5.2.3.2. Red núcleo: srsEPC

El módulo srsEPC, a pesar de ejecutarse como un solo binario, soporta los principales componentes del EPC: MME, HSS, SGW y PGW. Proporcionando una implementación ligera, pero funcional, de la red núcleo.

El HSS brindado almacena los datos de usuario en el formato `.csv`. Es importante que estos datos coincidan con los utilizados para grabar la tarjeta USIM. La información por UE registrada en el HSS, está relacionada con:

- Valor IMSI del UE.
- Por cada UE pueden ser seleccionados los algoritmos de autenticación: XOR o MILENAGE.
- Clave de UE (a partir de la que se derivan otras) almacenada en hexadecimal.
- Tipo de código del operador, ya sea OP u OPc.
- Código de operador (OP) o Código de operador cifrado (OPc), almacenado en hexadecimal.
- Campo de gestión de autenticación (AMF), almacenado en hexadecimal.
- Número de secuencia de UE (SQN) para la autorización de la autenticación.
- Identificador de clase de QoS para el servicio portador predeterminado del UE (CQI).
- Asignación de IP de forma dinámica o estática para UEs.

Para registrar un nuevo usuario en el HSS, se deben agregar al archivo `user_db.csv` los valores para los parámetros anteriores siguiendo un formato dado. En el fragmento 5.2, se muestra un fragmento del utilizado para este proyecto. En este caso se establece una dirección IP fija al UE para facilitar la interacción entre los equipos durante la fase de pruebas.

```
    #(ue_name),(algo),(imsi),(K),(OP/OPc_type),(OP/OPc_value),(AMF),(SQN)↩
    ,(QCI),(IP_alloc)
```

```
ue1,mil,901550123456789,00112233445566778899aabbccddeeff,opc,63↩
bfa50ee6523365ff14c1f45f88737d,9001,00000000173c,7,172.16.0.2
```

Fragmento 5.2: Configuración de HSS.

5.2.3.3. Equipo de usuario: srsUE

Por su parte, srsUE es el módulo que permite emular mediante software el UE de una red LTE. Al igual que srsENB, requiere de la interfaz de radio brindada por los SDR, admitiendo las familias Ettus USRP B2x0/X3x0, BladeRF y LimeSDR. Esta aplicación funciona de forma independiente por lo que es posible conectarla a una infraestructura LTE de estación base y red núcleo comerciales.

Entre sus prestaciones encontramos:

- Compatible de forma alineada con Release 10 de LTE, con algunas características de hasta la versión 15.
- Modos de transmisión 1 (antena única), 2 (diversidad de transmisión), 3 (CCD) y 4 (multiplexación espacial en bucle cerrado).
- Modos TDD y FDD para todos los anchos de banda (1.4, 3, 5, 10, 15 y 20 MHz).
- Frecuencias portadoras DL/UL configurables manualmente.
- Establecimiento de parámetros de USIM de forma manual o a través de un lector de USIM conectado a la PC.
- Capturas de paquetes de capa MAC y NAS.
- Soporte para eMBMS.

El archivo principal de configuración del srsUE es `ue.conf`, y a menos que se especifique otra ubicación en la línea de comandos, se carga de manera predeterminada desde la ubicación `~/.config/srslte/ue.conf`. Un fragmento del archivo de configuración utilizado se muestra en 5.3 y se describen sus principales parámetros.

```
[rf]
dl_earfcn = 3400
tx_gain = 47
#rx_gain = AGC
device_name = UHD
device_args = auto
#  Default for UHD: "recv_frame_size=9232, send_frame_size=9232,clock=↩
gpsdo"
```

```
#  For USRP B210 2x2 MIMO and >= 15 MHz USRP B210: "pass ↩
num_recv_frames=64,num_send_frames=64"
time_adv_nsamples = auto  #Default for B210: 100 samples
burst_preamble_us = auto  #Default for B210: 400 us
```

Fragmento 5.3: Configuración de srsUE.

Referente a la señal RF y el dispositivo de radio utilizado, los parámetros `dl_earfcn`, `dl_freq` y `ul_freq` establecen la frecuencia de UL y DL a utilizar. La ganancia de Tx y Rx es dada por `tx_gain` y `rx_gain`, respectivamente. Si el último de estos no está establecido o aparece comentado (como se muestra en el fragmento 5.3) se activa AGC para la Rx. La banda de frecuencias utilizada para el eNB debe coincidir con las definidas para el srsUE (pueden ser varias separadas por coma en `dl_earfcn`) para que este último sea capaz de detectar la celda e iniciar el proceso de *attach*, en este caso se utiliza el EARFCN 3400.

Otros de los parámetros mostrados son referentes a:

- `time_adv_nsamples`: cantidad de muestras para compensar el retraso de la señal generado desde la antena hasta la inserción de la marca de tiempo.
- `burst_preamble_us`: longitud del preámbulo para transmitir antes del inicio de la ráfaga.

La información respecto a la USIM personalizada, puede ser obtenida a partir de un lector de USIMs o desde el propio archivo de configuración, dependiendo del modo que se seleccione (`soft` o `pcsc`) (como se muestra en el fragmento 5.4). Es importante que los parámetros de la USIM se encuentren registrados correctamente en el HSS dentro de srsEPC, de lo contrario fallaría el proceso de autenticación del UE. En este caso se define Milenage como algoritmo de autenticación (ver sección 3.4.1).

```
[usim]
mode = soft
algo = mil
opc  = 63BFA50EE6523365FF14C1F45F88737D
k    = 00112233445566778899aabbccddeeff
imsi = 001010123456789
imei = 353490069873319
```

Fragmento 5.4: Configuración de USIM.

Como se muestra en el fragmento 5.5, se habilita la captura de paquetes de capa MAC en el formato .pcap y la interfaz gráfica provista por srsGUI. Se define el Release 10 a utilizar, así como la categoría 4 de UE como la predeterminada.

```
[pcap]
enable = false
nas_enable = false
[gui]
enable = true
```

```
[rrc]
ue_category        = 4
release            = 10
[phy]
```

Fragmento 5.5: Configuración de srsUE(continuación).

5.2.4. Programación de la tarjeta SIM

Al utilizar un teléfono móvil comercial es necesario contar con una tarjeta USIM compatible y personalizada con los parámetros definidos para la radio base y el HSS de la red. Para esto se utilizó el lector/quemador de tarjetas SIM compatible con 4G/FDD/LTE, mostrado en la figura 5.2. Con este dispositivo es posible escribir los parámetros como el ICCID (*Integrated Circuit Card Identifier*), IMSI, KI, OPC y OP, utilizando un software incluido (desde el sistema operativo Windows). Y sobre tarjetas USIM compatibles, también incluidas en el *kit*. Las tarjetas con las que se cuenta son reprogramables, y para facilitar su utilización con el lector/quemador, se utiliza un adaptador para SIMs de tamaños estándar, micro y nano.

Figura 5.2: Kit lector/quemador USIM.

5.2.5. iPerf

iPerf [1] es una herramienta multiplataforma útil para realizar mediciones que permitan caracterizar enlaces sobre el protocolo IP. Utilizando el enfoque cliente-servidor, genera un flujo de datos para obtener valores numéricos de la calidad del enlace. Ofrece

[1] Disponible en https://iperf.fr/.

gran flexibilidad para configurar los escenarios de prueba respecto al protocolo a utilizar, cronometraje y tamaño de los paquetes a transmitir. Por cada prueba se obtienen parámetros numéricos importantes como ancho de banda, pérdida de paquetes, entre otros.

De entre los protocolos de transporte a utilizar, se elige entre UDP y TCP. La principal diferencia entre ambos consiste en la confiabilidad de este último, debido a las confirmaciones de recepción (ACK) que solicita. De esta forma, el flujo de las comunicaciones en TCP puede ser más lento al incluir la espera de mensajes ACK y el reenvío de los paquetes perdidos. Por tanto, se decide usar UDP ya que permite un flujo regular de datos y un ligero ahorro de tiempo. Además, los mecanismos de evasión de congestión de TCP pueden resultar un atenuante cuando se quiere conocer, por ejemplo, el máximo ancho de banda del sistema.

5.2.6. Wireshark

Siendo uno de los analizadores de protocolos más utilizaos ampliamente en el mundo, Wireshark [1] se ha convertido en un estándar de facto cuando se trata de analizar a un nivel detallado los paquetes enviados y recibidos en una red determinada.

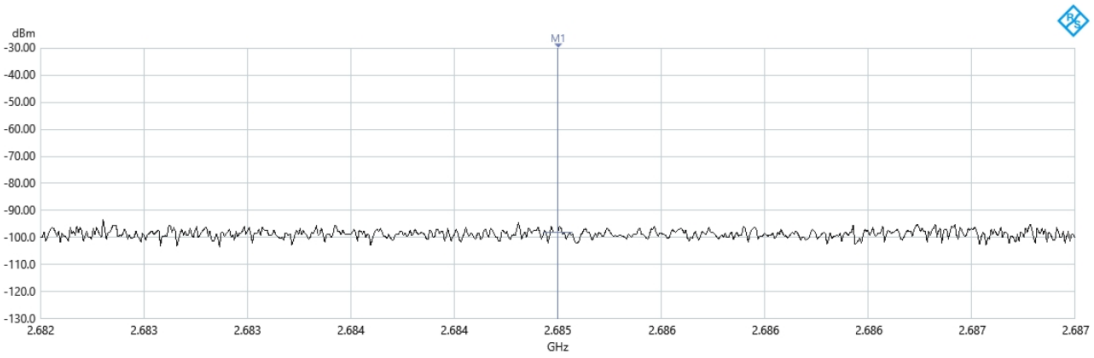
Es multiplataforma y permite la captura y filtrado, por ejemplo, por dirección IP y/o protocolos, de una gran cantidad creciente soportados (entre ellos LTE), para un posterior análisis offline. Ofrece además, soporte de descifrado para muchos protocolos, incluidos IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP y WPA/WPA2.

5.3. Selección de banda

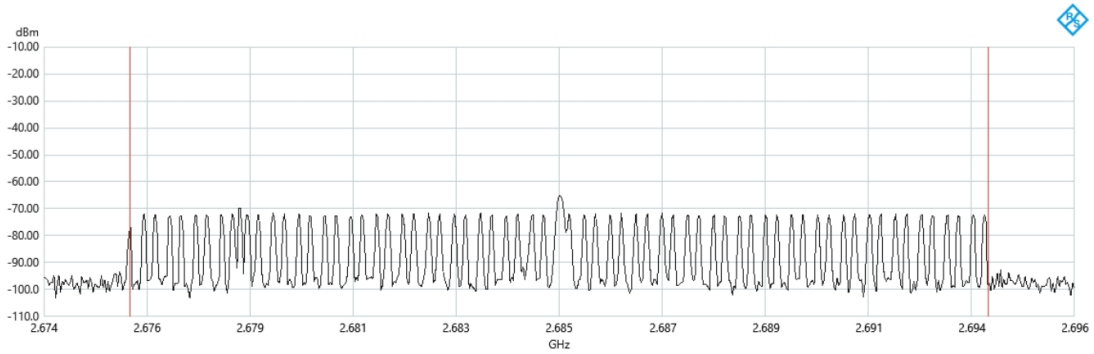
Las pruebas se realizaron empleando la banda 7 LTE-FDD (EARFCN=3400) con una frecuencia de enlace descendente de 2685 MHz y una frecuencia de enlace ascendente de 2565 MHz. Previamente se tomaron lecturas con el analizador de espectro modelo Rohde & Schwarz® FSH8 [64], determinándose que estas frecuencias centrales se encontraban desocupadas con un brecha mínima de 10 MHz a ambos lados. Esto con el objetivo de evitar interferencias al transmitir incluso con ancho de banda de 20 MHz. En la figura 5.3 se muestran lecturas de las frecuencias del canal de DL antes y durante la Tx de la radio base LTE.

Posteriormente, se comprueba el desempeño de las antenas con las que se cuenta (ver sección 5.1.2.2) en esta banda de frecuencia. En la figura 5.4 se muestran las pérdidas por retorno de la antena, o sea qué tanta energía es reflejada por la antena para un rango de frecuencias. Como se puede observar, para la frecuencia de 2.6 GHz se tiene

[1] Disponible en https://www.wireshark.org/.



(a)



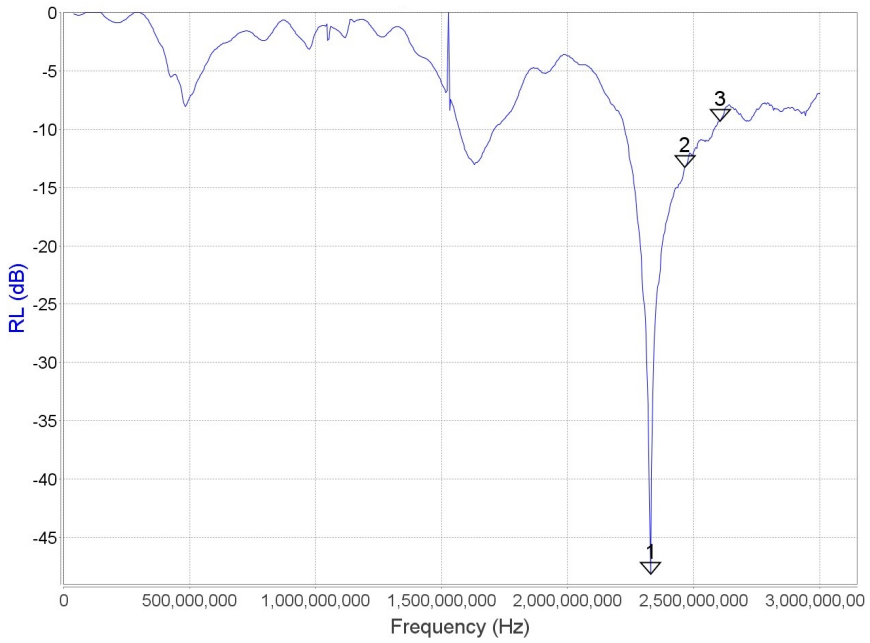
(b)

Figura 5.3: Lecturas del espectro para el canal de DL antes de Tx (a) y durante la Tx (b) usando 20 MHz de ancho de banda de LTE.

una pérdida por retorno de $-9.28\,\text{dB}$. Esto no es un valor deseable aunque tampoco es alarmante teniendo en cuenta la distancia a la que estarán ubicados los equipos en el ambiente indoor y las pruebas que se realizaron para calibrar la potencia de recepción deseada en ambos radios.

5.3.1. Consideraciones éticas

Dado que la banda de operación utilizada es comercial, se toman precauciones para evitar crear interferencias a operadores legítimos. Primeramente se comprueba que la banda se encuentra completamente desocupada. Seguidamente se configuran potencias de Tx mínimas de modo que el UE objetivo, ubicado a una distancia de 30 cm, pudiese conectarse a la radio base. Finalmente, se confirmó que ningún usuario legítimo intentó conectarse a la red LTE creada durante los experimentos.



Marker	Freq. (Hz)	RL (dB)	RP (°)	|Z| ()	Rs ()	Xs ()	Theta	SWR
1	2,268,208,850	-41.58	7.68	50.8	50.8	0.1	0.1	1.02:1
2	2,465,229,422	-10.79	-20.12	86.7	84.7	-18.3	-12.2	1.81:1
1-2	197,020,572	30.79	27.80	35.8	33.9	18.5	0.0	---
3	2,605,958,402	-8.42	-75.47	59.1	44.9	-38.5	-40.6	2.22:1

Figura 5.4: Pérdida de retorno vs frecuencia de antena VERT2450.

5.4. Conexión de los USRP

Una vez conectados los USRP por las interfaces correspondientes, se utiliza el comando `uhd_find_devices` para verificar que se hayan reconocido los dispositivos. Si todo sale bien, muestra información del dispositivo conectado incluyendo nombres de unidad, número de revisión y sensores disponibles en la placa base y placas secundarias.

Para evitar posibles desbordamientos y subdesplazamientos a altas velocidades de muestreo en la comunicación con los USRP conectados vía red, en este caso el X310, deben modificarse los tamaños máximos de búfer que vienen por defecto limitados en Linux para ahorrar recursos. Además, para alcanzar el máximo rendimiento, se recomienda establecer de forma manual el MTU de la interfaz a la que se encuentra conectado el radio en dependencia con la velocidad de Tx. En este caso al emplear la de 10 GigE se utiliza un valor de 9000. Para aplicar estas optimizaciones se ejecutan los siguientes comandos:

```
$ sudo sysctl -w net.core.rmem_max=33554432
$ sudo sysctl -w net.core.wmem_max=33554432
$ sudo ifconfig <interfaz> mtu 9000
```

5.5. Ejecución y confirmación de enlace

Una vez instalada la herramienta y configurados los parámetros correspondientes en cada uno de sus módulos, se procede a su ejecución. Se debe tener en cuenta que, al ejecutar las aplicaciones de los diferentes componentes LTE, debe hacerse con permisos de usuario root para que se puedan priorizar los procesos que se ejecutan en tiempo real y se permita la creación de interfaces virtuales. Los módulos srsENB y srsEPC pueden ser ejecutados en una misma PC, mientras que el srsUE deberá usarse en una máquina diferente.

Al iniciar, las aplicaciones tomarán la configuración predeterminada de los archivos correspondientes de cada módulo (expuestos anteriormente), ubicados en `~/.srs/`. La prueba de conectividad se lleva a cabo usando como equipos de usuario tanto el celular comercial como el terminal con la aplicación srsUE.

En la máquina *A* (ver sección 5.1.1) se despliegan los módulos del srsENB y srsEPC de la red LTE usando el USRP X310. Una vez conectado el dispositivo de radio se verifica la comunicación con el host usando el comando `uhd_usrp_probe`.

Seguidamente se ejecuta el comando `srsepc`, como se muestra en el fragmento 5.6. De esta forma se crea una interfaz virtual de red llamada `srs_spgw_sgi` con el IP 172.16.0.1, a la que pertenecerán los dispositivos de usuario de la red LTE que se conecten. Posteriormente se inicia el módulo srsENB como se muestra en el fragmento 5.7.

```
$ sudo srsepc
Built in Release mode using commit 5343b33f on branch master.
---  Software Radio Systems EPC  ---
Reading configuration file /home/srslte/.config/srslte/epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf901, MNC: 0xff55
SPGW GTP-U Initialized.
SPGW S11 Initialized.
```

```
SP-GW Initialized.
```

Fragmento 5.6: Ejecución de srsEPC.

```
$ sudo srsenb
Built in Release mode using commit 5343b33f on branch master.
---  Software Radio Systems LTE eNodeB  ---
Reading configuration file 123445/home/srslte/.config/srslte/enb.conf...
[INFO] [UHD] linux; GNU C++ version 7.4.0; Boost_106501; UHD_3.14.1.0-↩
    release
[INFO] [LOGGING] Fastpath logging disabled at runtime.
Opening USRP with args: type=b200,master_clock_rate=30.72e6
[INFO] [B200] Detected Device: B210
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Initialize CODEC control...
[INFO] [B200] Initialize Radio control...
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Asking for clock rate 30.720000 MHz...
[INFO] [B200] Actually got clock rate 30.720000 MHz.
Setting frequency: DL=2685.0 Mhz, UL=2565.0 MHz
[INFO] [B200] Asking for clock rate 11.520000 MHz...
Starting plot for worker_id=0
[INFO] [B200] Actually got clock rate 11.520000 MHz.
Setting Sampling frequency 11.52 MHz
==== eNodeB started ===
Type <t> to view trace
```

Fragmento 5.7: Ejecución de srsENB.

Por su parte, la máquina *B* (ver sección 5.1.1) es acondicionada con la aplicación srsUE y el radio USRP B210 para utilizarse como equipo de usuario. La ejecución se muestra en el fragmento 5.8.

```
$ sudo srsue
Reading configuration file /home/srslte/.config/srslte/ue.conf...
```

```
Built in Release mode using commit 5343b33f on branch master.
---  Software Radio Systems LTE UE  ---
Opening 1 RF devices with 1 RF channels...
[INFO] [UHD] linux; GNU C++ version 7.4.0; Boost_106501; UHD_3.14.1.0-↩
    release
[INFO] [LOGGING] Fastpath logging disabled at runtime.
Opening USRP with args: type=b200,master_clock_rate=30.72e6
[INFO] [B200] Detected Device: B210
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Initialize CODEC control...
[INFO] [B200] Initialize Radio control...
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Asking for clock rate 30.720000 MHz...
[INFO] [B200] Actually got clock rate 30.720000 MHz.
Waiting PHY to initialize...
Attaching UE...
Searching cell in DL EARFCN=3400, f_dl=2685.0 MHz, f_ul=2565.0 MHz
.................
```

Fragmento 5.8: Ejecución de srsUE.

A su vez, se creará una interfaz virtual llamada `tun_srsue` para sostener el enlace LTE a partir de la asignación de un IP en el rango 172.16.0.x, luego del proceso de registro.

5.5.1. Pruebas de conexión

Para probar la conectividad se utiliza el UE comercial con la USIM personalizada para realizar el proceso de attach y probar la conectividad a Internet. Al realizar la búsqueda de redes móviles de forma manual en el celular debe aparecer el PLMN-ID de la red creada, como muestra la figura 5.5.

Al seleccionar la red se lleva a cabo el proceso de attach y el UE establece una conexión RRC con el eNB y de tipo NAS con el MME. En la figura 5.6 se muestra un fragmento de las capturas, con la herramienta Wireshark, de los paquetes del protocolo NAS intercambiados entre el MME y el equipo de usuario emulado por la aplicación

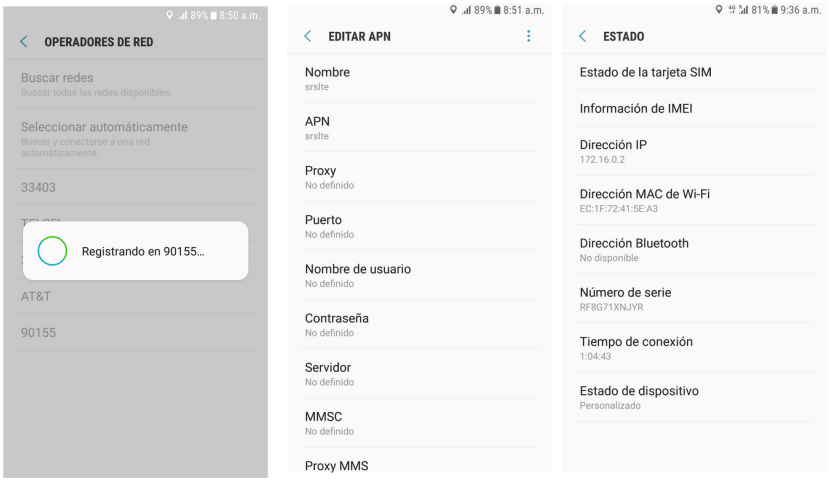


Figura 5.5: Conexión a red por UE comercial.

srsUE durante el registro.

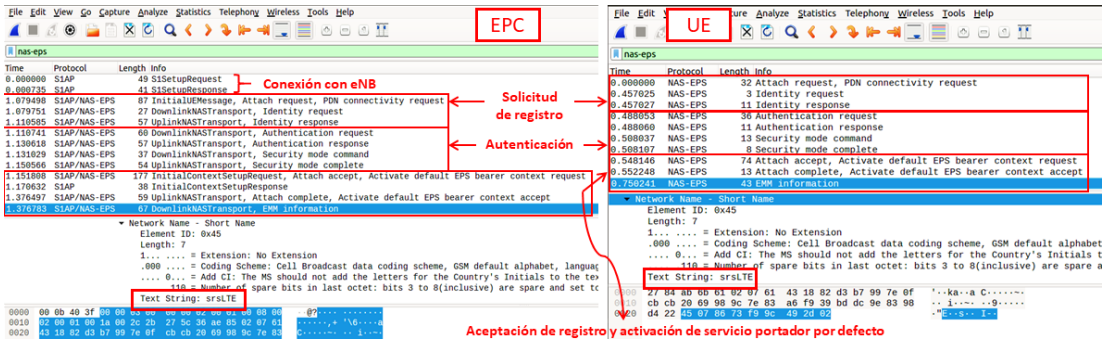


Figura 5.6: Captura de paquetes NAS entre UE y MME durante el registro del usuario.

Conectar el UE a Internet conlleva realizar un enmascaramiento de IP para hacer posible el reenvío de paquetes (*packet forwarding*) de la interfaz virtual de la red LTE hacia la que posea salida a Internet en la computadora. Para esto se ejecuta con permisos de administrador el script `srsepc_if_masq [out_interface]`, sustituyendo el texto entre corchetes por el nombre de esta última.

Además, en el propio celular se debe agregar un punto de acceso (APN, por sus siglas en inglés), como se muestra en la figura 5.5. De esta forma se modifica el servicio portador EPS por defecto con los parámetros de terminación del túnel GTP en el contexto asociado al servicio portador del usuario en cuestión en el SGW, permitiendo el acceso a Internet.

5.6. Análisis teórico de la tasa de Tx de datos

Para evaluar el desempeño mostrado por la herramienta srsLTE en las pruebas realizadas, se toma como base un análisis teórico del throughput alcanzable para el DL y UL, sobre una configuración dada. Con ese objetivo, en esta sección se ofrecerán dos enfoques: uno realizando cálculos sobre el conocimiento general que se tiene sobre el estándar LTE y otro más preciso utilizando datos de especificaciones técnicas (TS, por sus siglas en inglés) dadas por 3GPP.

En los siguientes subapartados se describen y muestra un ejemplo de cada uno y posteriormente se completa una tabla con los valores obtenidos para las configuraciones a desarrollar.

5.6.1. Enfoque de conocimiento general

Este enfoque toma en cuenta conocimiento general del estándar LTE, parte del cual fue expuesto en el Capítulo 3. Específicamente referente a la distribución de recursos de capa física (ver sección 3.6). El ejemplo a desarrollar, tanto para el enlace DL como UL, se basa en una configuración de throughput pico de LTE FDD utilizando 20 MHz de ancho de banda, una modulación de 64-QAM y la utilización de MIMO 2x2 por multiplexación espacial (aplica para el DL). De esta forma, el número de RE en un subframe (1 ms) puede ser calculado de la siguiente manera:

$$RE_{PorSubframe_{20\,\mathrm{MHz}}} = 7_{smbolosOFDM} * 12_{subportadoras} * 100_{RBs} * 2_{slots} = 16800 \quad (5.1)$$

Ancho de banda	QPSK		16-QAM		64-QAM	
	SISO	MIMO[1]	SISO	MIMO[1]	SISO	MIMO[1]
1.4	1.51	3.02	3.02	6.05	4.54	9.07
3	3.78	7.56	7.56	15.12	11.34	22.68
5	6.30	12.60	12.60	25.20	18.90	37.80
10	12.60	25.20	25.20	50.40	37.80	75.60
15	18.90	37.80	37.80	75.60	56.70	113.40
20	25.20	50.40	50.40	100.80	75.60	151.20

(a)

Ancho de banda	QPSK	16-QAM	64-QAM[2]
1.4	1.512	3.024	4.536
3	3.78	7.56	11.34
5	6.3	12.6	18.9
10	12.6	25.2	37.8
15	18.9	37.8	56.7
20	25.2	50.4	75.6

(b)

Tabla 5.1: Throughput máximo teórico (Mb/s) para DL (a) y UL (b) usando enfoque general.

[1]Se refiere al modo MIMO2x2 por multiplexación espacial.

[2]El modo 64-QAM solo aplica para UE Categoría 5 u 8.

Según el cálculo anterior se estarán enviando un total de 16 800 RE por cada milisegundo, que es la duración de un subframe. Considerando que se utiliza una modulación de 64-QAM, donde cada símbolo OFDM contiene 6 bits de información, la máxima tasa de Tx (sin tener en cuenta codificación) está dada por la siguiente expresión:

$$Throughput\_Max_{20MHz/64QAM} = 16800_{RE/ms} * 6_{bits/symbol} = 100.8_{Mb/seg} \quad (5.2)$$

Para lograr un valor más realista es necesario tener en cuenta la información que se envía para sincronización y control de la señal. La cual se considera es de aproximadamente el 25 % del total del ancho de banda. Actualizando el valor obtenemos una tasa

pico aproximada de 75.6 Mb/s. Si a esto le adicionamos una configuración de MIMO 2x2 por multiplexación espacial, donde es posible utilizar ambas antenas para enviar información simultanea por caminos virtuales independientes, obtenemos un total de 151.2 Mb/s para el enlace descendente.

Respecto al enlace ascendente, donde no se tiene en cuenta una configuración multiantena para la Tx, se obtiene un throughput pico de 75.6 Mb/s. En las tablas 5.1 se muestran los resultados calculados usando este enfoque, variando ancho de banda y esquema de modulación.

5.6.2. Enfoque orientado a especificaciones técnicas de 3GPP

Este enfoque ofrece una aproximación más exacta del throughput máximo ya que está basado en el tamaño de los bloques de transporte definidos por el 3GPP, tanto para DL como para UL. En este caso, primeramente se hace uso de las tablas 3.5 y 3.6 para obtener los índices de TBS (I_{TBS}) según el MCS utilizado, para DL y UL, respectivamente. Seguidamente se obtiene el número de bloques de recursos (N_{PRB}) según el ancho de banda a utilizar, como muestra la tabla 3.2.

Teniendo el TBS (I_{TBS}) y (N_{PRB}), se puede obtener la cantidad de bits que se transportan por cada PRB. Para esto, se realiza un mapeo por fila y columna utilizando las tablas ofrecidas en TS 36.306 del 3GPP para el Release 10 de LTE [42] a partir de la página 35.

En el caso de MIMO2x2 por multiplexación espacial, tal como indica el estándar, se sigue el siguiente procedimiento:

- Para $1 \leq N_{PRB} \leq 55$, el TBS se obtiene igualmente mapeando en la tabla correspondiente pero utilizando los valores ($I_{TBS}, 2 * N_{PRB}$).
- Para $55 \leq N_{PRB} \leq 110$, se obtiene de la propia tabla un valor de referencia `TBS_L1` que se utiliza para mapear el valor `TBS_L2` en otra de las tablas ofrecidas por el estándar. El TBS entonces está dado por el valor `TBS_L2`.

Ancho de banda	Modo de Tx	QPSK MCS=9	16-QAM MCS=16	64-QAM MCS=28
1.4	SISO	0.936	1.8	4.392
	MIMO[1]	1.864	3.624	8.76
3	SISO	2.344	4.584	11.064
	MIMO[1]	4.776	9.144	22.152
5	SISO	4.008	7.736	18.336
	MIMO[1]	7.992	15.264	36.696
10	SISO	7.992	15.264	36.696
	MIMO[1]	15.84	30.576	75.376
15	SISO	11.832	22.92	55.056
	MIMO[1]	16.416	45.352	110.136
20	SISO	15.84	30.576	75.376
	MIMO[1]	31.704	61.664	149.776

(a)

Ancho de banda	QPSK MCS=10	16-QAM MCS=20	64-QAM[2] MCS=28
1.4	1.032	2.6	4.392
3	2.664	6.456	11.064
5	4.392	10.68	18.336
10	8.76	21.384	36.696
15	12.96	32.856	55.056
20	17.568	43.816	75.376

(b)

Tabla 5.2: Throughput máximo teórico (Mb/s) para DL (a) y UL (b) usando enfoque basado en TS.

[1]Se refiere al modo MIMO2x2 por multiplexación espacial.

[2]El modo 64-QAM solo aplica para UE Categoría 5 u 8.

Una vez obtenido el TBS, y teniendo en cuenta que un bloque de recurso es transmitido por cada TTI de 1ms, se puede calcular la tasa de bits por segundo ofrecida por el sistema. A modo de ejemplo, al igual que en el epígrafe anterior se obtendrá mediante este enfoque un valor de throughput pico. Dado por la configuración: LTE-FDD de 20 MHz usando 64-QAM y MIMO 2X2 con multiplexación espacial, para DL y UL.

En este sentido, se considera un MCS de 28, el cual arroja $I_{TBS} = 26$, tanto para DL como UL. Entonces, dado el ancho de banda mencionado de 20 MHz, con $N_{PRB} = 100$, en caso de UL se obtiene directamente $\texttt{TBS}_{(26,100)} = 75376$. Lo cual indica una tasa pico para el canal de subida de 75.376 Mb/s.

En el canal de bajada, al utilizar MIMO 2x2, se mapea y obtiene el valor $\texttt{TBS\_L1}_{(26,100)} = 75376$. Luego, usando este último en la tabla correspondiente, se ubica el TBS final dado por $\texttt{TBS\_L2}_{75376} = 149776$. Esto quiere decir que para la configuración dada el throughput pico para DL es de aproximadamente 149.98 Mb/s. Como se puede observar la tasa obtenida es bastante parecida a la de 151.2 Mb/s lograda según las proposiciones del enfoque anterior. En las tablas 5.2 se muestran los resultados para UL y DL obtenidos usando este enfoque, variado ancho de banda y esquema de modulación.

5.7. Descripción de escenarios

El diseño de los escenarios de prueba, partiendo de los objetivos trazados para esta investigación, pretende analizar el comportamiento de la red implementada con srsLTE y equipos USRP. El análisis del desempeño se basa principalmente en la obtención de parámetros que caracterizan la señal de radio, la latencia del enlace y la tasa de Tx de datos en los canales de UL y DL. Este último indicador se compara con los resultados teóricos obtenidos en la sección anterior.

Para llevarlo a cabo se despliega primeramente un escenario cableado donde se varían los esquemas de modulación y codificación utilizando cada ancho de banda definido para LTE. Posteriormente, se expone un segundo escenario sobre la interfaz aire Over-the-Air (OTA) y utilizando como UEs un celular comercial y el USRP B210. Este se enfocado en comprobar la versatilidad de la herramienta y comparar el desempeño del USRP frente al equipo celular. Por cuestiones de compatibilidad con el teléfono móvil, en el escenario OTA se emplean solamente los anchos de banda de 5 MHz y 10 MHz.

5.7.1. Metodología de pruebas

En cada prueba, una vez completado el proceso de registro del UE en la red, se genera tráfico durante 100 segundos, lo cual se considera tiempo suficiente respecto a

la duración del frame de LTE (10 ms). Se adquieren datos sobre calidad del enlace de radio y para el cálculo del Round-Trip Delay Time (RTT). Además, sobre las tasas de Tx por cada ancho de banda de LTE y para cada canal.

Para la generación de tráfico se emplea la herramienta iperf. Teniendo en cuenta las prestaciones computacionales de los equipos con los que se cuenta, se configura para crear 20 flujos paralelos de datos sobre el protocolo UDP. Todos con un ancho de banda de 1 Gbit/seg para evitar que este indicador acote la capacidad alcanzable. Utilizando un script (ver Apéndice A), se repite esta operación durante cinco iteraciones; ya que se encontró que para esa cantidad de iteraciones se alcanzaban valores estables. Luego, se promedian los resultados por cada indicador para lograr una lectura más confiable

Para las mediciones del enlace descendente se ejecuta el servidor iPerf en el UE, mientras el cliente se inicia desde el equipo portador del eNB. En caso inverso, para las mediciones del UL se ejecuta el proceso cliente y servidor en los nodos del UE y eNB, respectivamente. En el fragmento 5.9 se muestran los comandos utilizados.

```
$ iperf -s -u -P 20 -i 1  -f m
```

(a)

```
$ iperf -c ip-server -p 5001 -u -t 100 -P 10 -b 1000M
```

(b)

Fragmento 5.9: Comandos de iPerf para servidor (a) y cliente (b).

5.7.2. Configuración general

En las pruebas de conexión cableada los radios se conectan directamente utilizando un cable coaxial (RG-58), conectores SMA y un atenuador de−30 dB; para evitar daños a los dispositivos RF. De esta forma, los dispositivos USRP emulando el eNB y el UE usan un par de conexiones directas cableadas para simular una situación ideal de los canales de Tx y Rx, dada la resistencia a interferencias que ofrece el cable coaxial. Los atenuadores se conectan más cercanos al puerto de Rx. Se estima que la atenuación introducida por los cables es mínima en esta configuración (aproximadamente −1 dB).

Por su parte, en el modo OTA se utilizan las antenas descritas en el apartado 5.1.2.2 ubicando los equipos a una distancia de 30 cm.

Respecto a la ganancia de transmisión, se necesita especificar el indicador `tx_gain` a través de la plataforma de software. Para obtener el valor adecuado para este parámetro se establece un enlace de 5 MHz entre el UE y la radio base y se ejecutan peticiones de

tipo ICMP utilizando el comando `ping`. De esta forma, existiendo conexión y un flujo de datos con paquetes de poca carga útil (64 bytes), se observa el CQI reportado por la estación base. Luego, comenzando con valores pequeños se aumenta gradualmente la ganancia de Tx hasta obtener el CQI máximo de 15. Esto indica que se puede usar un esquema de modulación de 64-QAM y la más eficiente tasa de codificación en el DL, logrando al mismo tiempo un BLER igual o menor al 10 %.

Se observa que el CQI buscado se presenta en promedio cuando el UE reporta un RSRP aproximado de −88 dB. Además, una vez resuelta la ganancia adecuada, el CQI no muestra variación significativa para el resto de los anchos de banda. Por tanto se decide mantener constante el valor de `tx_gain` dentro de cada entorno. Los valores obtenidos se muestran en la tabla 5.3.

	Cableado	OTA
X310	1	8
B210	47	55

Tabla 5.3: Valores de ganancias de Tx en software utilizados por los USRP en cada entorno.

Capítulo 6

Resultados

En el presente capítulo se argumentan y despliegan los escenarios de prueba de la plataforma. Especificándose configuraciones y optimizaciones realizadas, acordes a cada uno de ellos, para lograr el funcionamiento estable del sistema. También se especifica la metodología a seguir, incluyendo la automatización de la captura de los datos de salida, para lo cual se elaboran scripts interpretados en Bash. Finalmente, se muestran y analizan los resultados obtenidos caracterizándose los entornos primeramente teniendo en cuenta el comportamiento de la señal de radio, la latencia y la tasa de bits por segundo lograda con el enlace.

6.1. Gestión de datos de pruebas

6.1.1. Captura de registros

Con el objetivo de automatizar y sincronizar la captura de logs se elaboran programas interpretados en Bash. En la tabla 6.1 se muestran los parámetros requeridos para su ejecución dependiendo del diseño de cada prueba.

Para limitar el número de pruebas a realizar por cada ancho de banda, se decide tomar los MCS con paso 4. De esta forma se toman 8 valores para el DL (0,4,8,12,16,20,24 y 28) y 6 valores para UL. Debido a que este canal no utiliza el último esquema de modulación (64-QAM), se omiten los MCS 24 y 28 (ver sección 3.6).

El script (ver Apéndice A) se debe ejecutar de forma simultánea en cada host y lleva a cabo la siguiente secuencia de pasos lógicos:

1. Lectura de los parámetros de configuración pasados como argumentos del programa.

Parámetro	Descripción	Opciones
n_prb	Cantidad de PRB	6|15|25|50|75|100
mcs	Esquema de modulación	0-28
tx_gain	Ganancia de Tx definida por software	1 | 8 para X310 47 | 55 para B210
iperf_mode	Modo de ejecución de iPerf	*server* | *client*
srslte_mode	Módulos srsLTE a iniciar	*ue* | *enb_epc*
measurement_mode	Ejecutor de medición	*active* | *inactive*

Tabla 6.1: Anchos de banda de celdas compatibles con LTE.

2. Creación de una carpeta destino para los logs a generar, nombrada según los parámetros.
3. Desactivación del modo de ahorro de los procesadores para evitar el escalado de frecuencia y lograr el máximo desempeño de estos.
4. Modificación de los archivos de configuración de él/los módulo(s) srsLTE a ejecutar en correspondencia con la configuración dada y creación y establecimiento de la carpeta destino para los logs.
5. Ejecución de él/los módulo(s) correspondiente(s) según el valor de `srslte_mode`.
6. Si `iperf_mode=server`, se ejecuta el proceso del servidor iPerf que va a observar la llegada de los paquetes desde el cliente.
7. Utilizando el comando *ping* se verifica la conectividad entre srsEPC y srsUE.
8. Luego, si `measurement_mode=active`:

   I Ejecución de comando `ping` al ip remoto para medir RTT.

   II Inicialización de un ciclo de 5 iteraciones donde:

      1) Se verifica el mantenimiento de la conexión y se obtiene del SO información previa sobre la cantidad de paquetes transmitidos y recibidos tanto por el host local como el remoto.
      2) Se ejecuta el cliente iPerf en el nodo correspondiente durante 100 segundos.
      3) Al terminar, se verifica el mantenimiento de la conexión y se obtiene igualmente información posterior brindada por el SO.

4) Con los valores registrados se calcula la cantidad de paquetes transmitidos por el cliente y recibidos efectivos en el servidor durante la iteración.

III Al finalizar las 5 iteraciones se exportan estadísticas generales sobre la Tx y Rx de paquetes para un futuro procesamiento.

9. Por último se restablece la configuración predeterminada de rendimiento de los CPU.

El host designado como `measurement_mode=active` será el encargado de capturar, no solo los datos generados por srsLTE, sino también por el comando `ping` para muestreo de Round-Trip Delay Time (RTT) y las estadísticas emitidas por el SO sobre la Tx y Rx de paquetes en las interfaces del host local y remoto. Para la interacción con el host remoto se establece una conexión mediante el protocolo SSH [1]. Se definió el envío de trazas de la herramienta cada 5 segundos, para evitar afectar los valores de latencia a obtener. De esta forma por cada iteración se obtienen alrededor de 24 muestras de métricas y 120 por cada prueba.

6.1.2. Procesamiento y presentación

Una vez capturada la información de las pruebas se ejecuta otra aplicación (ver Apéndice B) para clasificar y descartar archivos de pruebas incompletas o con parámetros diferentes a los definidos anteriormente.
Seguidamente, se utiliza otro script (ver Apéndice C) para parsear los registros de cada prueba y obtener los indicadores a analizar, normalizar sus valores en algunos casos y exportar la información deseada a archivos de datos. Estos archivos son usados como entrada de la aplicación *Gnuplot* [2] para ser graficados.

En la presentación de los datos se utilizan gráficas de tres y dos dimensiones. Para el análisis específico de la latencia se utiliza la función de distribución acumulada (CDF). Se plantea el uso de este modelo estocástico ya que facilita la caracterización de la señal a partir de su comportamiento estadístico.

6.2. Despliegue de escenario cableado

Para el análisis de este escenario se registraron 96 pruebas a la plataforma, según las siguientes variaciones: 8 (MCSs) * 6 (anchos de banda LTE) * 2 (canales de DL y UL). En la figura 6.1 se muestra una foto tomada durante la ejecución de una de las prueba.

[1] Ver `https://www.openssh.com/`.
[2] Ver `http://www.gnuplot.info/`.

Figura 6.1: Escenario cableado.

Al inicio de cada prueba, una vez establecido el enlace LTE, se realizan algunas calibraciones manuales para estabilizar la conexión del nodo ejecutando srsUE con la radio base. Los parámetros involucrados son `freq_offset` y `time_adv_nsamples`.

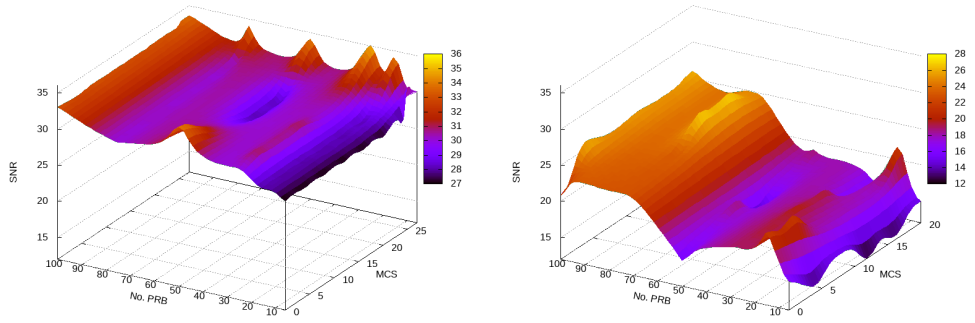
En el caso del primero, se utiliza para contrarrestar el desfase de frecuencia teniendo en cuenta el valor `cfo` del canal de DL reportado por el UE en sus trazas. Por su parte, `time_adv_nsamples` se utiliza para mejorar la sincronización entre las interfaces de radio debido a un leve retraso que se genera justo después de agregarles la marca de tiempo que indica el momento de transmisión y hasta el momento real en que se transmite. El valor a obtener se busca definiéndolo primeramente en 0. Luego, una vez concretada la conexión, se observa el indicador `ta_us` enviado por las trazas del usuario. Entonces, el número de muestras que se deben poner por adelantado de acuerdo a la frecuencia de muestreo se calcula de la siguiente forma $time\_adv\_nsamples = F_s * ta\_us$.

6.2.1. Comportamiento de la señal

Para el análisis de la señal se tiene en cuenta principalmente los iniciadores SNR, BLER (ver sección 3.10) y su repercusión en el CQI reportado por el UE.

El SNR es inversamente proporcional a la cantidad del ruido en el sistema. Es por esto que utilizar modulaciones de orden superior, sin que se cuente con un SNR óptimo, puede provocar mayor probabilidad de error de bit. Se debe a que al existir más símbolos en la constelación y encontrarse más próximos entre sí, es más probable que se detecte erróneamente un símbolo en lugar de otro, debido a interferencias o ruido.

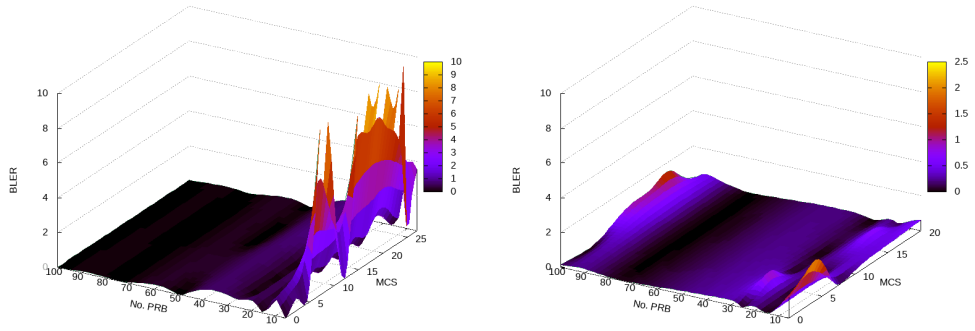
En la figura 6.2, se da una representación de los valores obtenidos para el SNR



(a) (b)

Figura 6.2: Comportamiento del SNR en el escenario cableado para DL (a) y UL (b).

según lo reportado por el UE durante las mediciones del DL y por el eNB durante las saturación del enlace UL. Se aprecia un comportamiento similar para ambos canales a partir de que este indicador es bajo al inicio y aumenta ligeramente junto con la cantidad de PRB. Igualmente, se realza apropiadamente al utilizarse un esquemas de modulación de 64-QAM en el DL (para UL solo se maneja hasta 16-QAM). Como es posible notar, el SNR del UL tiene un mayor intervalo de fluctuación que el del DL y se encuentra por debajo de este por lo general. Aun así se logran valores aceptables para una buena calidad del canal.



(a) (b)

Figura 6.3: Comportamiento del BLER en el escenario cableado para DL (a) y UL (b).

Así mismo el BLER, como se muestra en la figura 6.3, expide valores más elevados al utilizarse el menor ancho de banda de 1.4MHz. Debido a las bajas tasas de Tx que este soporta y la saturación de tráfico en el canal, la pérdida de paquetes por error

en bloques repercute en mayor medida. Aun así en ambos canales este indicador se mantiene por debajo del recomendable 10 %.

Por su parte el término CQI permite caracterizar la calidad de la señal desde el punto de vista del UE (ver sección 3.7). En condiciones normales, a partir de este se mapea un MCS adecuado. Sin embargo, esta función se encuentra deshabilitada intencionalmente para observar el comportamiento del sistema.

Como muestra la figura 6.4, los valores CQI reportados por el UE sobre la calidad del enlace descendente se ven afectados proporcionalmente a la disminución del SNR y al aumento del BLER. Este efecto es una parte clave del funcionamiento del mecanismo de codificación y modulación adaptativa de LTE. Permitiendo al eNB administrar los recursos de radio para mantener un BLER objetivo menor del 10 %.

Es importante destacar sobre este parámetro que, a pesar de que fue calibrado en un inicio para mantenerse en el valor máximo de 15 (en condiciones de poca carga en la red), comenzó a variar a partir del aumento del tráfico y en correspondencia con los indicadores vistos anteriormente.

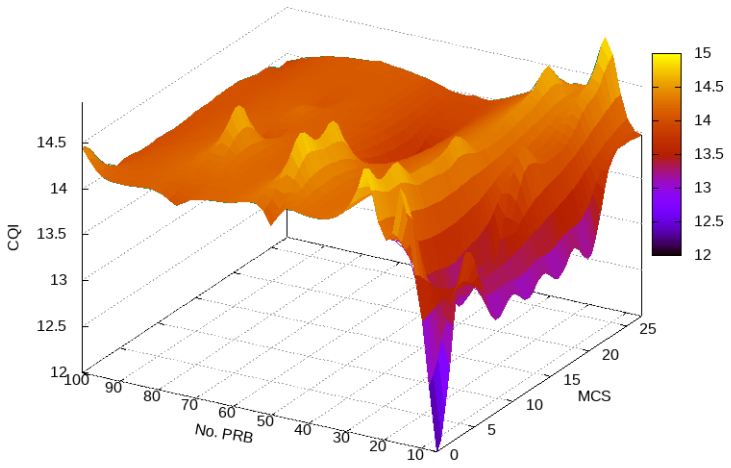


Figura 6.4: Comportamiento del CQI en el escenario cableado para DL.

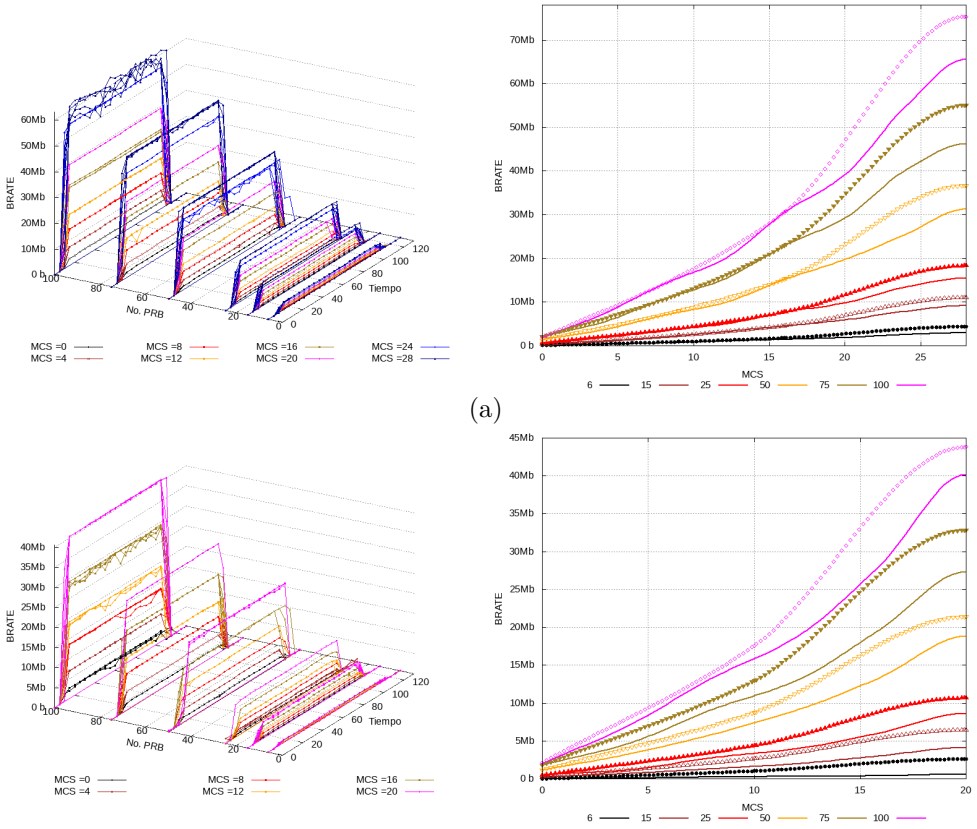


Figura 6.5: Comportamiento de la tasa de bits [b/s] en el escenario cableado para DL (a) y UL (b).

6.2.2. Tasa de Tx de datos

Para analizar la transmisión de datos sobre el canal se tienen en cuenta tanto las métricas de trazabilidad enviadas por la herramienta como las estadísticas sobre las interfaces de red dadas por el SO. Para esto último, se obtiene justo antes de iniciar y terminar la prueba, la cantidad de bytes enviadas por el nodo en modo cliente y los recibidos por el nodo en modo servidor como muestra el fragmento 6.2.2 del script elaborado (ver Apéndice A). Siguiendo esta metodología se puede obtener la cantidad de bytes que fue Rx por el nodo servidor, siendo esta el throughput efectivo de la red.

```
tx_bits_byiteration=$( $client_mode cat /sys/class/net/$client_adapter↩
/statistics/tx_bytes )
rx_bits_byiteration=$( $server_mode cat /sys/class/net/$server_adapter↩
/statistics/rx_bytes )
```

En la figura 6.5 se muestra, para DL y UL respectivamente el comportamiento de la tasa de Tx de datos en las pruebas para los escenarios cableados y OTA. En la gráfica en 3D se representan los datos tomados de los logs de srsLTE. En la 2D se puede observar con líneas de colores por cada número de PRB, el resultado de aplicar la función de suavizado `mcsplines` de Gnuplot a los datos obtenidos de las interfaces de red. Devolviendo así una interpolación cúbica que conserva la monotonicidad y convexidad de los puntos de datos originales. Igualmente, se muestra con puntos discontinuos del mismo color respectivo el comportamiento teórico esperado según los valores obtenidos en la sección 5.6.2.

En cada prueba, una vez que se ejecuta el cliente iPerf el tráfico de datos comienza a aumentar rápidamente hasta alcanzar la velocidad máxima. Luego el rendimiento se mantiene relativamente estable en el caso del escenario cableado, fluctuando más principalmente en los últimos MCS (>= 20). Finalmente, el flujo de datos comienza a disminuir al terminar las iteración y en la medida que se desocupan los buffers de los nodos.

Como se puede apreciar, el rendimiento del sistema aumenta en correspondencia del MCS, y dependiendo de la cantidad de PRB utilizados, puede llegar a incrementarse en la misma medida. Por ejemplo, al aumentar en el doble el no. PRB de 25 a 50, se obtiene aprox. el doble de throughput. Por otra parte, los valores reales obtenidos persiguen la curva teórica esperada, aunque se mantienen por debajo de esta (ver figura 6.5). Esto se explica considerando que no todos los recursos de radio establecidos se utilizan para la Tx de datos de un usuario en los enlaces UL/DL.

6.2.3. Latencia del enlace

Muchas de las aplicaciones y tecnologías que van ganando popularidad en la actualidad, por ejemplo la realidad aumentada (RA), vehículos autónomos, audio/vídeo llamadas de alta definición, etc.; son sensibles al retardo. Por tanto, la reducción de la latencia en la interfaz de radio es una de las principales preocupaciones en el desarrollo de LTE. En este sentido, por ejemplo, se adopta la utilización de un TTI corto de 1 ms.

Para analizar la latencia del enlace entre la estación base y el UE se utiliza el comando `ping` para obtener valores de RTT. Este indicador muestra el tiempo de ida y vuelta e incluye contribuciones de latencia en el plano del usuario, tiempos de procesamiento de la aplicación y demoras en la red de transporte.

Las aplicaciones sensibles a la latencia también agradecen la poca variabilidad del RTT o *jitter*. En este caso, no se puede medir con precisión usando el comando `ping` dado que las respuestas de eco dependen del planificador del sistema operativo del host remoto, lo cual introduce jitter en sí.

Al utilizar el comando indicado, por defecto se envían y reciben paquetes de tipo ICMP con un *payload* de 64 bytes. En este caso, como dirección de host remoto se utiliza la del UE (172.16.0.0.2) o la de la interfaz del EPC (172.16.0.1) dependiendo de dónde se haya ejecutado. Para evitar que la saturación de tráfico generado por el iPerf afecte en estas mediciones, se pausa la ejecución del script (usando el comando `sleep`) los primeros 20 s antes de iniciar esta aplicación. Considerando 5 iteraciones, contamos con una muestra total de 100 valores por cada prueba. En la tabla 6.2 se muestran un resumen sobre el estudio de la latencia del enlace con mayor, menor y promedio de RTT, mostrado en ms (milisegundos).

		Cableado		
No. PRB	Host remoto	Min.	Ave.	Máx.
6	UE -> EPC	26.9	38	49.8
6	EPC -> UE	14.6	37	49.1
15	UE -> EPC	12.1	26	37.7
15	EPC -> UE	11.7	30	46.9
25	UE -> EPC	13.8	26	36.9
25	EPC -> UE	14.1	28	45.1
50	UE -> EPC	9.08	26	37.0
50	EPC -> UE	14.3	26	36.3
75	UE -> EPC	12.6	26	36.9
75	EPC -> UE	13.8	27	37.0
100	UE -> EPC	16.8	27	39.5
100	EPC -> UE	16.9	27	37.5

Tabla 6.2: Estadísticas sobre la latencia del enlace.

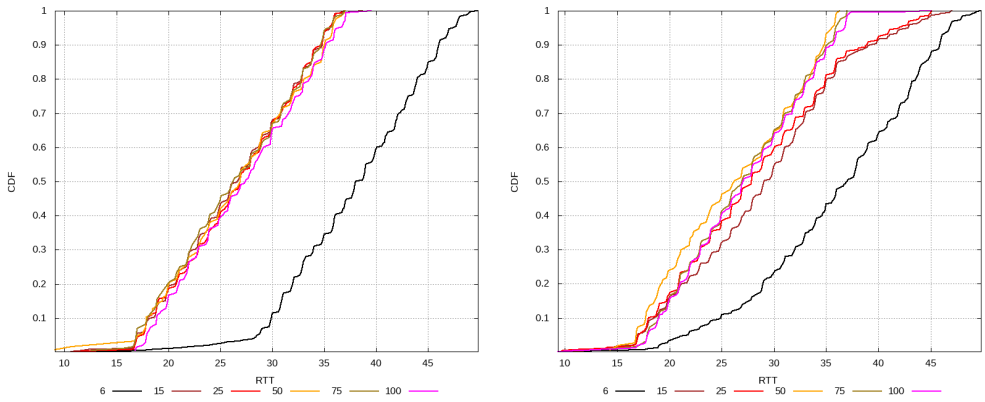
Para los sistemas IMT-Advanced, se define una latencia *one-way* de plano de usuario alcanzable de 10 ms en condiciones de poca carga (una red con un usuario y un flujo de datos) y para paquetes IP pequeños (por ejemplo, con encabezado pero sin carga útil) para ambos enlaces descendentes y ascendente [65]. En la práctica los niveles suelen oscilar en el orden de los 5 ms y 50 ms por cada canal.

Como se muestra en los resultados del estudio del RTT, los valores obtenidos oscilan entre los rangos esperados. En algunos casos la latencia es menor de la esperada alcanza-

ble (20 ms para *round-trip*). También se tienen valores altos probablemente provocados por errores en el canal de radio que se traducen en retransmisiones. Esta función es manejada en LTE a través del mecanismo HARQ a nivel de capa MAC y típicamente el tiempo de ejecución RTT del procedimiento de retransmisión es de 8 ms [12].

El RTT depende de varios factores. Dado que en el escenario los nodos se encuentran directamente conectados, se pueden despreciar algunos como son la distancia y/o cantidad de saltos para alcanzar al nodo destino. Respecto al uso de los USRP, estos a su vez introducen latencia al emplear un sistema de bus para transferir las muestras de la interfaz de radio al procesador. Incluso, el uso de un procesador de propósito general para el procesamiento de la señal agrega latencias adicionales que no se encuentran en radios convencionales. Sin embargo, dada las altas tasas de Tx que usan los medios de interconexión de los radios a la PC y el uso de CPUs de nueva generación a altas velocidades de procesamiento, se puede estimar que la latencia agregada por estos se encuentra en el orden de los microsegundos [57].

En la figura 6.6, se representa el comportamiento obtenido para el RTT en el DL y UL mediante la CDF. En ambos canales se obtiene un comportamiento bastante parecido y a su vez por cada MCS, excepto para el ancho de banda de 1.4 MHz. En este caso se aprecia la influencia de otro de los factores que degradan al RTT y es el ancho de banda de los nodos. Dado que en este caso se cuenta con solamente 6 PRB para Tx, de los cuales 2 son asignados al canal de control y señalización.



(a) (b)

Figura 6.6: Comportamiento del RTT en escenario cableado para DL (a) y UL (UL).

6.3. Despliegue de escenario OTA

En este escenario se utiliza un teléfono celular comercial y el USRP B210 para conectarse a la radio base (uno a la vez) utilizando la interfaz aire y a una distancia de 30 cm. De esta forma se pretende comparar el desempeño del USRP respecto a un dispositivo LTE comercial y a su vez el escenario OTA respecto al cableado expuesto en la sección anterior. Para las pruebas se utilizan los anchos de banda de 5 MHz y 10 MHz y el canal de DL. A pesar de que se intentó con anchos de banda superiores para obtener mayor throughput, se nota que el teléfono celular no era capaz de encontrar la radio base al emplear la banda 7.

En la figura 6.7 se muestra una foto tomada durante el despliegue de este escenario utilizando, en este caso, el B210 como UE.

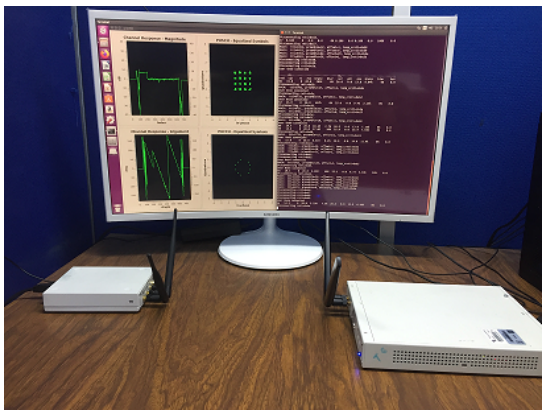
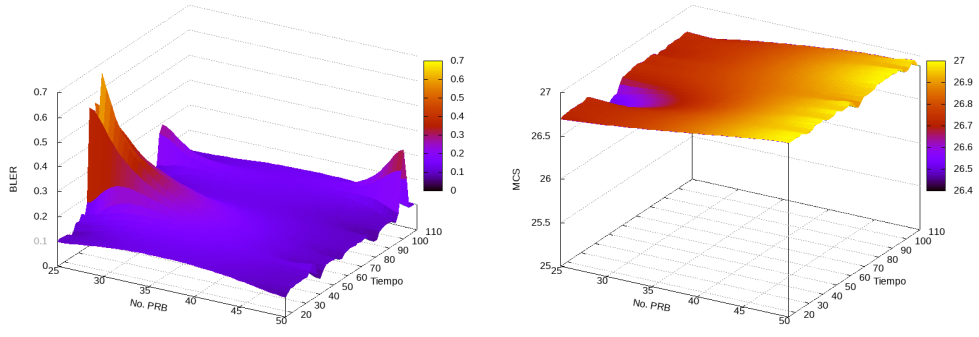


Figura 6.7: Escenario OTA

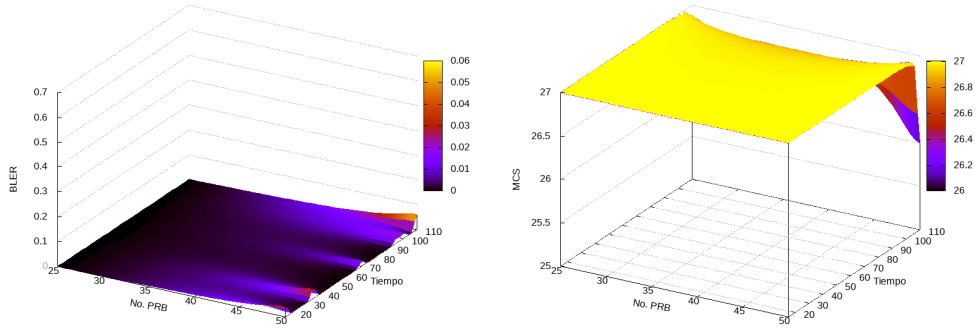
6.3.1. Comportamiento de la señal

Para obtener lecturas parciales en las pruebas realizadas, se tienen en cuenta solamente los parámetros reportados por la srsENB sobre el canal de DL: BLER y MCS. Debido a que el teléfono móvil no arroja logs procesables con el programa elaborado, como es el caso de srsUE.

La figura 6.8 muestra el comportamiento del BLER y MCS obtenido en el despliegue OTA. Como se puede observar, se obtienen valores bajos (siempre menores al 10 %) para el BLER, con leves fluctuaciones obtenidas en las mediciones con el celular. Esto último se refleja en el MCS escogido por el eNB para el ancho de banda correspondiente.



(a)



(b)

Figura 6.8: Comportamiento de BLER y MCS en el escenario OTA usando celular (a) y USRP (b) como UE.

6.3.2. Throughput

En la figura 6.9, se muestran los valores promediados obtenidos por cada iteración en cuanto a la tasa de Tx de datos para el canal de DL, agregándose como referencia el resultado pico teórico esperado. Los valores reales obtenidos se mantienen por debajo del máximo teórico calculado para estos anchos de banda, respecto a ello cabe destacar que en las pruebas realizadas el MCS máximo ofrecido por el srsENB es de 27 y no de 28 (límite superior definido por el estándar para el Release LTE tratado).

Por otro lado, se observa que la señal se mantiene estable en el tiempo de iteración. Apreciándose que para el ancho de banda de 5 MHz (25 PRB) el celular mantiene un desempeño medio respecto al B210 en los escenarios cableados y OTA. Sin embargo para un ancho de banda mayor, en este caso de 10 MHz (50 PRB), se nota un leve mejor desempeño del equipo UE comercial respecto a los USRP.

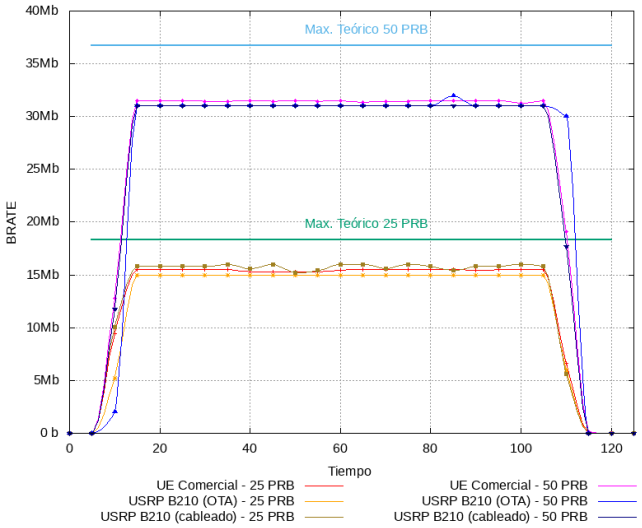


Figura 6.9: Comportamiento de la tasa de bits en escenario usando celular y USRP como UE.

Capítulo 7

Conclusiones

En el presente trabajo se indagó sobre el principio de operación de los USRP, así como de las ventajas que conlleva su utilización alternativa, dadas principalmente, por la gran versatilidad de estos y sus bajos costos respecto a equipos RF específicos para estándares como LTE. Precisamente, siendo este último otra de las grandes motivaciones para esta investigación, se indaga sobre las bases de su funcionamiento y su uso sobre plataformas SDR. A partir de ello se investigó sobre trabajos relacionados y plataformas existentes. Seleccionándose el software srsLTE para el despliegue dado su creciente prestigio en la comunidad de desarrollo en torno al tema.

Para favorecer el entendimiento de dicha plataforma, se ofrece una descripción partiendo del propio análisis de las tecnologías LTE y SDR. Abarcando la estructura y funcionamiento de la propia herramienta en sí. Además, se exponen los detalles necesarios para su instalación, configuración y despliegue.

Con el objetivo de evaluar el desempeño del sistema obtenido, se diseñan escenarios de prueba tanto cableados como OTA, indicándose las tecnologías y herramientas a utilizar. Se describe la metodología seguida para la optimización, captura y presentación de datos arrojados una vez implementados los escenarios de prueba.

En el primero de estos, se muestran y analizan los resultados obtenidos acerca de la señal radioeléctrica, las tasas de Tx y la latencia del enlace LTE al variar el esquema de codificación y modulación (MCS) por cada ancho de banda LTE. También se despliega un escenario OTA utilizando como UEs tanto un celular comercial como un USRP B210. En este, igualmente se analiza el comportamiento de la señal de radio y de las tasas de Tx obtenidas, permitiendo una comparativa entre ambos dispositivos.

A partir del trabajo realizado con los USRP, principalmente en las pruebas a altos anchos de banda (>10MHz), se observa que luego de un uso prolongado (3 horas aprox.) comienzan a funcionar inestables, por lo que hay que dejarlos reposar. Intuitivamente se puede explicar que el sobrecalentamiento provoca fallas en el oscilador y por tanto

en el enlace de radio. Debido a esto se decide tomar un descanso mínimo de 15 minutos entre cada prueba, en el cual se desconectan y ventilan los radios para que recuperen su temperatura.

Como se puede observar en los estudios de la señal de radio y el throughput para el enlace cableado, no se obtiene un comportamiento lo suficientemente idóneo a lo esperado. Como se mencionaba anteriormente, en algunas iteraciones se obtienen desconexiones súbitas y variaciones del SNR. Este comportamiento se hizo más evidente durante las pruebas de búsqueda de la ganancia de Tx ya que se nota que cuando este parámetro sobrepasa los 30dB (aprox.), para el caso del X310, el valor real tiende a mantenerse constante y aún más comienza a afectarse el SNR indicando interferencias. Dado lo anterior, se puede afirmar que los USRP utilizados son factibles para el despliegue temporal de escenarios de prueba y el estudio del estándar. Sin embargo, no se recomienda un uso exhaustivo y continuo de los mismos al emplearlos, por ejemplo, para desplegar un servicio LTE continuo.

Finalmente se concluye que al comprender el funcionamiento de LTE y los equipos USRP con los que se cuenta, fue posible realizar el despliegue y análisis una red basada en estas tecnologías, sobre la plataforma srsLTE. El resultado del desarrollo práctico y teórico realizado, al emplear un enfoque SDR basado en procesadores de propósito general, ofrece una útil y rentable herramienta para estudios más específicos en el área de las comunicaciones inalámbricas donde se requiera una red LTE funcional. Además, sirve como base para futuras investigaciones basadas en mejoras y/o evoluciones al estándar, así como en conceptos emergentes de 5G como C-RAN (real-time Cloud infrastructures RAN) y NB-IoT (Narrowband Internet of Things).

7.1. Recomendaciones y trabajos futuros

Siendo srsLTE una herramienta de software muy completa es posible el despliegue de otros escenarios de prueba variando otros parámetros del estándar como: esquema de Tx (para utilización de MIMO), utilización de eMBMS (Evolved Multimedia Broadcast Multicast Services), categoría de UE, Release LTE a emplear, entre otras.

Además, la existencia de otras herramientas de software que permiten desplegar el estándar LTE sobre dispositivos SDR, como openLTE y OAI, motivan la realización de un estudio comparativo del desempeño de estas.

Apéndice A

Script para ejecutar pruebas y capturar información de logs

```
#!/bin/bash

setting_ue_args(){
  printf "\nSetting UE config...\n"
   sed -i "s/.*tx_gain =.*/tx_gain = ${tx_gain}/
   s/^filename =.*/filename = ${folder_location//\//\\/}\/${file_name}_ue.↩
    log/
   s/.*release .*=.*/release = 10/
   s/.*ue_category .*/ue_category = 4/
   s/.*metrics_csv_enable .*/metrics_csv_enable = true/
   0,/^filename =.*/ s//filename = ${folder_location//\//\\/}\/${file_name↩
    }_ue.pcap/
   s/^nas_enable =.*/nas_enable = true/
   s/^nas_filename =.*/nas_filename = ${folder_location//\//\\/}\/${↩
    file_name}_ue_nas.pcap/
   s/.*metrics_csv_filename =.*/metrics_csv_filename = ${folder_location↩
    //\//\\/}\/${file_name}_ue_metrics.csv/" ${srslte_config_folder}ue.↩
    conf
   egrep 'tx_gain =|nas_enable =|filename =|ue_category =|release =|↩
    metrics_csv_enable =|metrics_csv_filename =|time_adv_nsamples =|↩
    burst_preamble_us =|freq_offset =' ${srslte_config_folder}ue.conf
}
```

```
setting_enb_args(){
   echo "Setting eNB config at "$srslte_config_folder
   sed -i "s/^n_prb =.*/n_prb = ${n_prb}/
   s/.*tm .*/tm = ${transmission_mode}/
   s/.*tx_gain =.*/tx_gain = ${tx_gain}/
   s/.*$mcs_type .*/$mcs_type = ${modulation_scheme}/
   s/filename =.*/filename = ${folder_location//\//\\/}\/${file_name}_enb.↩
    log/
   0,/filename =.*/ s//filename = ${folder_location//\//\\/}\/${file_name}↩
    _enb.pcap/
   s/.*metrics_csv_enable .*/metrics_csv_enable = true/
   s/.*metrics_csv_filename =.*/metrics_csv_filename = ${folder_location↩
    //\//\\/}\/${file_name}_enb_metrics.csv/" ${srslte_config_folder}enb.↩
    conf
   egrep "n_prb |tm =|tx_gain =|$mcs_type =|filename =|↩
    metrics_csv_filename =|time_adv_nsamples =|burst_preamble_us =|↩
    freq_offset =" ${srslte_config_folder}enb.conf
}

setting_epc_args(){
  echo "Setting EPC config at "$srslte_config_folder
  sed -i "s/filename =.*/filename = ${folder_location//\//\\/}\/${↩
    file_name}_epc.log/
  0,/filename =.*/ s//filename = ${folder_location//\//\\/}\/${file_name}↩
    epc.pcap/" ${srslte_config_folder}epc.conf
  egrep 'filename =' ${srslte_config_folder}epc.conf
}

connecting_remote(){
while true ; do
  if ping -q -c 1 -W 1 $remote_ip >/dev/null; then
    printf "\nEPC connected!!!\n" ; return 0
  else
    echo "No EPC connection" ; sleep 1s
  fi
```

```
done
}

beging_performance_mode(){
  echo "Setting maximun performance mode..."
#  cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
    for file in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do ↩
    echo "performance" > $file; done
#  cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
}

beging_powersave_mode(){
  echo "Returning to power save mode..."
#  cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
    for file in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do ↩
    echo "powersave" > $file; done
#  cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
}

HELP='Args needed: \n -n_prb  >> number_of_PRB: 6,15,25,50,75 or 100 \n -↩
    mcs >> modulation_scheme : [0-28] \n -tm >> transmission_mode : [1-5]
 -uecat >> ue_category : [3-5] \n -tx_gain >> tx_gain : [1-80] \n -im >> ↩
    iperf_mode : client or server \n -remote_ip  >>  : remote_ip \n -↩
    srslte_mode >>  : mode ue or enb_epc \n \n'

args_count=0
for i in "$@" ; do
case $i in
    -h*|--help*)
    printf "$HELP"
    shift
    ;;
    -n_prb=6|-n_prb=15|-n_prb=25|-n_prb=50|-n_prb=75|-n_prb=100*)
    n_prb="${i#*=}";((args_count++))
    shift
    ;;
```

```
    -tm=[1-5]*)
    transmission_mode="${i#*=}";((args_count++))
    shift
    ;;
    -uecat=[3-5]*)
    ue_category="${i#*=}";((args_count++))
    shift
    ;;
    -tx_gain=[1-80]*)
    tx_gain="${i#*=}";((args_count++))
    shift
    ;;
    -im=client|-im=server)
    iperf_mode="${i#*=}";((args_count++))
    shift
    ;;
    -srslte_mode=enb_epc|-srslte_mode=ue)
    srslte_mode="${i#*=}";((args_count++))
    shift
    ;;
    -measure_mode=active|-measure_mode=inactive)
    measure_mode="${i#*=}";((args_count++))
    shift
    ;;
    -mcs=*)
    modulation_scheme="${i#*=}";((args_count++))
    shift
    ;;
    --default)
    DEFAULT=YES
    shift
    ;;
esac ; done
echo "n_prb                 = ${n_prb}"
echo "modulation_scheme     = ${modulation_scheme}"
echo "transmission_mode     = ${transmission_mode}"
```

```
echo "ue_category           = ${ue_category}"
echo  "tx_gain              = ${tx_gain}"
echo "iperf_mode            = ${iperf_mode}"
echo "srslte_mode           = ${srslte_mode}"
echo "measure_mode          = ${measure_mode}"

if [[ -n $1 ]]; then
    echo "One or some arguments have error..."
    printf "$HELP"; exit 0
elif [[ $args_count != 8 ]]; then
    echo "Found $args_count and $(( 8 - $args_count )) argument(s) are ↩
    missing..."
    printf "$HELP"; exit 0
fi

##### CONSTANTS #####
srslte_config_folder=/home/srslte/.config/srslte/
now=$( date +%d-%m_%T );  test_time=100 ; no_iterations=5 ; no_threads=20 ↩
    ; mcs_type="pdsch_mcs"
file_name=$measure_mode.$srslte_mode.$n_prb.$modulation_scheme.↩
    $transmission_mode.$ue_category.$tx_gain.$iperf_mode.$no_iterations.↩
    $no_threads..$now
echo "Folder name: $file_name [measure_mode.srslte_mode.number_of_PRB.↩
    modulation_scheme.transmission_mode.ue_category.tx_gain.iperf_mode.↩
    cant_iteraciones.no_threads] + $now"
mkdir $file_name;
folder_location=$(pwd)/$file_name
autolog_file=$folder_location/autoapp_log; exec > >(tee -i $autolog_file);↩
     exec 2>&1
echo $folder_location > ~/.srslte_logs

play_radios(){
if [ $srslte_mode = "ue" ]; then
  local_ip=172.16.0.2 ; remote_ip=172.16.0.1
  local_adapter=tun_srsue ; remote_adapter=srs_spgw_sgi
  setting_ue_args
```

```
  gnome-terminal --geometry=101x28+0-0 --hide-menubar -- sh -c "srsue ${↩
    srslte_config_folder}ue.conf | tee ${autolog_file}_ue_console_$1.log"
  sleep 4s
else
  local_ip=172.16.0.1 ; remote_ip=172.16.0.2
  local_adapter=srs_spgw_sgi ; remote_adapter=tun_srsue
  if [ $iperf_mode = "server" ]; then mcs_type="pusch_mcs" ; fi
  setting_epc_args
  setting_enb_args
  gnome-terminal --geometry=101x28+0-0 --hide-menubar -- sh -c "srsepc ${↩
    srslte_config_folder}epc.conf | tee ${autolog_file}_epc_console_$1.log↩
    "
  sleep 3s
  gnome-terminal --geometry=102x28+1000-0 --hide-menubar -- sh -c "srsenb ↩
    ${srslte_config_folder}enb.conf | tee ${autolog_file}_enb_console_$1.↩
    log"
fi
}
play_radios "0"
remote_ssh_connection='ssh root@'$remote_ip
beging_performance_mode
## BEGINNING iPERF SERVER AND CLIENT
if [ $iperf_mode = "server" ]; then
      client_mode=$remote_ssh_connection ; server_mode=""
      client_adapter=$remote_adapter
      server_adapter=$local_adapter ; server_ip=$local_ip
          ####  RUNNING SERVER  ####
      gnome-terminal -- sh -c "echo BEGINNING iPerf server $( date +"%T-%N↩
    " )... ; iperf -s -u -P $(( $no_iterations * 20 )) -i 1  -f m | tee ${↩
    autolog_file}_iperf_server.log"
else
      server_mode=$remote_ssh_connection ; client_mode=""
      server_adapter=$remote_adapter ; server_ip=$remote_ip
      client_adapter=$local_adapter ;
fi
```

```
connecting_remote
if [ $measure_mode = "active" ]; then
#### RUN PING ####
gnome-terminal --hide-menubar -- sh -c "ping ${remote_ip} | tee ${↩
    autolog_file}_ping_remote.log"
####  INITIALIZE GLOBAL VARS  ####
ave_tx_bits_byiteration=0; ave_rx_bits_byiteration=0; ave_bits_lost=0
for i in $( seq 1 $no_iterations ); do
  ####  INITIALIZE TEMPORAL VARS  ####
  tx_bits_byiteration=0 ; rx_bits_byiteration=0
  ####  VERIFYING CONNECTION  ####
  connecting_remote
  #### PING MEASUREMENT ####
  sleep 20s
  #### ADAPTERs INFO BEFORE  ####
  echo "CLIENT ADAPTERs INFO BEFORE at $( date +"%T-%N" )"
  tx_bits_byiteration=$( $client_mode cat /sys/class/net/$client_adapter/↩
    statistics/tx_bytes )
  echo "SERVER ADAPTERs INFO BEFORE at $( date +"%T-%N" )"
  rx_bits_byiteration=$( $server_mode cat /sys/class/net/$server_adapter/↩
    statistics/rx_bytes )
  echo "RUNNING CLIENT at $( date +"%T-%N" )"
  ####  RUNNING CLIENT  ####
  $client_mode iperf -c ${server_ip} -p 5001 -u -t ${test_time} -P ↩
    $no_threads -b 1000M -f m &
  sleep ${test_time}s
  ####  VERIFYING CONNECTION  ####
  connecting_remote
  ####  ADAPTERs INFO AFTER   ####
  echo "CLIENT ADAPTERs INFO AFTER at $( date +"%T-%N" )"
  tx_bits_byiteration_aux=$( $client_mode cat /sys/class/net/↩
    $client_adapter/statistics/tx_bytes )
  echo "SERVER ADAPTERs INFO AFTER at $( date +"%T-%N" )"
  rx_bits_byiteration_aux=$( $server_mode cat /sys/class/net/↩
    $server_adapter/statistics/rx_bytes )
  tx_bits_byiteration=$(($tx_bits_byiteration_aux-$tx_bits_byiteration))
```

```
  rx_bits_byiteration=$(($rx_bits_byiteration_aux-$rx_bits_byiteration))
  ####  CALCULING THROUGHPUTs ####
  echo "CALCULING THROUGHPUTs at $( date +"%T-%N" )"
  ave_tx_bits_byiteration=$(($ave_tx_bits_byiteration+$tx_bits_byiteration↩
    ))
  ave_rx_bits_byiteration=$(($ave_rx_bits_byiteration+$rx_bits_byiteration↩
    ))
  lost_bits=$(($tx_bits_byiteration-$rx_bits_byiteration))
  ave_bits_lost=$(($lost_bits+$ave_bits_lost))
  echo "**************************************************"
  echo "Enviado [Mbps] (client) total de iteracion $i: $(( ↩
    tx_bits_byiteration / 1000000 ))"
  echo "Recibido [Mbps] (server) efectivo de iteracion $i: $(( ↩
    rx_bits_byiteration / 1000000 ))"
  echo "**************************************************"
  echo "Enviado (client) total de iteracion $i: $tx_bits_byiteration"
  echo "Recibido (server) efectivo de iteracion $i: $rx_bits_byiteration"
  echo "Bites no recibidos en iteracion $i: $lost_bits"
done
echo "TEST FINISHED AT $( date +"%T-%N" )"
echo "**************************************************"
echo "Enviado [Mbps] promedio x iteteracion (client): $((↩
    $ave_tx_bits_byiteration/$i/1000000)) "
echo "Recibido [Mbps] promedio x iteteracion (server): $((↩
    $ave_rx_bits_byiteration/$i/1000000)) "
echo "Promedio throughput [Mbps] total: $(($ave_tx_bits_byiteration/↩
    $test_time/$i/1000000)) "
echo "Promedio throughput [Mbps] efectivo: $(($ave_rx_bits_byiteration/↩
    $test_time/$i/1000000)) "
echo "Promedio bites no recibidos: $(($ave_bits_lost/$i)) "
echo "**************************************************"
echo "Enviado promedio x iteteracion (client): $(($ave_tx_bits_byiteration↩
    /$i)) "
echo "Recibido promedio x iteteracion (server): $((↩
    $ave_rx_bits_byiteration/$i)) "
echo "Promedio throughput total: $(($ave_tx_bits_byiteration/$test_time/$i↩
```

```
    )) "
echo "Promedio throughput efectivo: $(($ave_rx_bits_byiteration/$test_time↩
    /$i)) "

else
  i=1 ; continuar="ini"
  while [ $continuar != "exit" ]; do
    echo "Restarted radios? (yes or exit)... " ; read continuar
    if [ $continuar = "yes" ]; then
      play_radios $i ; ((i++))
    fi
  done
fi
beging_powersave_mode
spd-say 'Finished test...'
echo "Finished test..."
exit 0
```

Apéndice B

Script para filtrar pruebas con lecturas exitosas

```
#!/bin/bash

srsLTE_enb_measurement(){
if [ -s autoapp_log_enb_console_* ] ; then
for log in autoapp_log_enb_console_*.log ; do
  sub_console=$( echo $log | cut -d'_' -f5 | cut -d'.' -f1 )
  measure_array=($( egrep '^[0-9]' autoapp_log_enb_console_${sub_console}.↩
    log ))
  i=0 ; j=5 ; count=0
  for elt in "${measure_array[@]}"; do
    ((i++))
    if [ $i -gt 11 ]; then i=0 ; ((j+=5)) ; ((count++)) ; fi
    if [ $j -gt $time_test ] ; then answ="BUENA LECTURA EN $(pwd)" ; break↩
    ; fi
  done
done
fi
}

srsLTE_ue_measurement(){
if [ -s autoapp_log_ue_console_* ] ; then
```

```
for log in autoapp_log_ue_console_*.log ; do
  sub_console=$( echo $log | cut -d'_' -f5 | cut -d'.' -f1 )
  measure_array=($( egrep '^ [0-9]' autoapp_log_ue_console_${sub_console}.↩
    log ))
  i=0 ; j=5 ; count=0
  for elt in "${measure_array[@]}"; do
    ((i++))
    if [ $i -gt 13 ]; then i=0 ; ((j+=5)) ; ((count++)) ; fi
    if [ $j -gt $time_test ] ; then answ="BUENA LECTURA EN $(pwd)" ; break↩
    ; fi
  done
done
fi
}

############## VARS ##############
local_folder=$(pwd)
t_mode=1
tx_gain=*
time_test=120
mcs_order=( "-1" "0" "4" "8" "12" "16" "20" "24" "28" )
ue_category=*
no_hilos=*
no_iterations=*
iperf_mode=*
modulation=*
n_prb=*
srslte_mode=*
measure_mode=*

############## MAIN ##############
    for f in * ; do
      if [[ -s $f ]]; then
      srslte_mode=$(echo $f | cut -d'.' -f2);
      iperf_mode=$(echo $f | cut -d'.' -f8);
      if [[ "( "ue" "enb_epc" )" =~ "$srslte_mode" ]] && [[ "( "server" "↩
```

```
client" )" =~ "$iperf_mode" ]]; then
    echo ${srslte_mode}
    n_prb=$(echo $f | cut -d'.' -f3); echo n_prb $n_prb
    tx_gain=$(echo $f | cut -d'.' -f7); echo tx_gain $tx_gain
    no_hilos=$(echo $f | cut -d'.' -f10 ); echo no_hilos $no_hilos
    measure_time=$(echo $f | cut -d'.' -f12 | cut -c 1-11); echo measure_time $measure_time
    measure_mode=$(echo $f | cut -d'.' -f1); echo measure_mode $measure_mode
    modulation_scheme=$(echo $f | cut -d'.' -f4)
    echo "$modulation_scheme $tx_gain $no_hilos "
    if [[ "${mcs_order[@]}" =~ "${modulation_scheme}" ]] && ([ ${tx_gain} -eq 75 ] || [ ${tx_gain} -eq 55 ]) && [ $no_hilos -ge 20 ] ; then
      if ([ "$srslte_mode" == "enb_epc" ] && [ "$iperf_mode" == "client" ]) || ([ "$srslte_mode" == "ue" ] && [ "$iperf_mode" == "server" ])  ; then test_mode="DL"
      else test_mode="UL"; fi
      echo "MCS >> $modulation_scheme"
      cd $f
      ##### srsLTE TRACE MEASUREMENT
      answ=""; if [ "$srslte_mode" = "ue" ]; then srsLTE_ue_measurement
      else srsLTE_enb_measurement ; fi
      cd ..
      if [[ -n $answ ]]; then
        n_prb=$(echo $f | cut -d'.' -f3)
        echo $answ
        mkdir -p good_antenna_MIMO/${n_prb}_PRB/${test_mode}/${modulation_scheme}/${measure_time}/
        mv -v $f/ good_antenna_MIMO/${n_prb}_PRB/${test_mode}/${modulation_scheme}/${measure_time}/
      else
        if [[ ! -n "incompletes_antenna" ]]; then mkdir incompletes ; fi
        mv -v $f/ incompletes/ ;
```

```
        fi
      else
      if [[ ! -n "trash_antenna" ]]; then mkdir trash ; fi
        mv -v $f/ trash/
      fi
    fi
  fi
  done
```

Apéndice C

Script para parsear y graficar resultados

```
#!/bin/bash

standardized_to_unit(){
  if [[ $1 =~ "k" ]];  then
     echo "${1//k/}*1000" | bc
  elif [[ $1 =~ "M" ]] || [[ $1 =~ "m" ]]; then
      echo "${1//[M|m]/}*1000000" | bc
  else
      echo "${1//k/}" | bc
  fi
}

adapterstatistics_measurement(){
  if [ -s autoapp_log ] ; then
  echo " >>> Scraping autoapp_log"
  bytes_received_stream=( $(awk '/^Recibido \(server\)/{print $7}' ↩
    autoapp_log) )
  for elt in "${bytes_received_stream[@]}"; do
      bytes_received=$elt
      if [ -n "$bytes_received" ] && [ "$bytes_received" -eq "↩
    $bytes_received" ] 2>/dev/null; then
        bits_received=$( echo "$bytes_received*8/100" | bc -l )
        echo -e "$mcs_e \t $bits_received " >> $path_to_folder/${↩
    folder_name}_efective_brate_adapterstatistics
```

```
      fi
  done
  fi
}

ping_measurement(){
  if [ -s autoapp_log_ping_remote.log ] ; then
  echo " >>> Scraping $(find autoapp_log_ping_remote.log)"
  icmp_stream=($( awk '/^[0-9]/{print $7}' autoapp_log_ping_remote.log ))
  ave_icmp_time=0 ; ping_count=0
  for i in $( seq 0 120 ); do
      icmp_time=$( echo "${icmp_stream[$i]//time=/}" )
      if (( $(echo "$icmp_time < 50" | bc -l ) )); then
        ave_icmp_time=$( echo "$ave_icmp_time+$icmp_time" | bc -l )
        echo -e "$mcs_e \t $icmp_time" >> $path_to_folder/${folder_name}↩
    _remote_rtt_icmp
        ((ping_count++))
      fi
  done
  printf "
  ++++++++++++++++++++++++++++++ AVE ICMP TIME MCS = $mcs_e ↩
    ++++++++++++++++++++++++++++++
  $( echo "$ave_icmp_time/$ping_count" | bc) \n" >> $path_to_folder/${↩
    folder_name}_ave_traces
  fi
}

enb_csv_metrics(){
  if [ -s *_enb_metrics.csv ] ; then
  echo " >>> Scraping $(find *_enb_metrics.csv)"
  sed -i "s/;/$(printf "\t")/g" *_enb_metrics.csv
  metrics_stream=($( egrep '^[0-9]' *_enb_metrics.csv | cut -f-4 -d$'\t' )↩
    )
  ave_dl_brate_metric=0 ;ave_ul_brate_metric=0 ; time=0 ;metric_time=0; ↩
    max_ul_brate=0; max_dl_brate=0
  if [[ $iperf_mode == "server" ]]; then test_index=3 # MONITORING ↩
```

```
  DOWNLINK
else  test_index=2 #MONITORING UPLINK
fi
iteration_count=0
threshold_test=$( standardized_to_unit ${measure_array[$test_index]} )
it_beginning_flag=-1000
row_lengh=4
flag_rollback=false
flag_playtrace=false
for ((k=0; k<${#measure_array[@]}; k++)); do
  elt=${measure_array[$k]}
  if [ $(($k%$row_lengh)) -eq $test_index ]; then threshold_test=$( ↩
  standardized_to_unit $elt ); fi
  if [ $(($iteration_count-$it_beginning_flag)) -ge $((↩
  $no_iterations_test)) ]; then flag_playtrace=false; fi
  if (( $(echo "$threshold_test >= $threshold_brate" | bc -l ) )) && [ ↩
  $flag_playtrace = "false" ] ; then
      flag_rollback=true
      flag_playtrace=true
      it_beginning_flag=$iteration_count
      sub_time_iter=0
  elif [ $flag_playtrace = "true" ]; then
      case $(($k%$row_lengh)) in
        2)
          dl_brate_metric=$( standardized_to_unit $elt )
          if (( $(echo "$dl_brate_metric > 1" | bc -l ) )); then
          ave_dl_brate_metric=$( echo "$ave_dl_brate_metric+↩
  $dl_brate_metric" | bc -l )
          echo -e "$sub_time_iter\t $dl_brate_metric" >> $path_to_folder↩
  /${mcs_e}/${time_folder}/${folder_name}_csv_dl_brate_enb_metric
          if (( $(echo "$dl_brate_metric > $max_dl_brate" | bc -l ) )); ↩
  then max_dl_brate=$dl_brate_metric; fi
          fi
          shift
          ;;
        3)
```

```
            ul_brate_metric=$( standardized_to_unit $elt )
            if (( $(echo "$ul_brate_metric > 1" | bc -l ) )); then
            ave_ul_brate_metric=$( echo "$ave_ul_brate_metric+↩
    $ul_brate_metric" | bc -l )
            echo -e "$sub_time_iter\t $ul_brate_metric" >> $path_to_folder↩
    /${mcs_e}/${time_folder}/${folder_name}_csv_ul_brate_enb_metric
            if (( $(echo "$ul_brate_metric > $max_ul_brate" | bc -l ) )); ↩
    then max_ul_brate=$ul_brate_metric; fi
            fi
            shift
            ;;
        esac
        if [ $(($k%$row_lengh)) -eq 0 ]; then
          ((sub_time_iter+=5)) ; ((iteration_count++)); fi
        fi
        if [ $flag_rollback = "true" ]; then
          k=$(( $k - ($k%$row_lengh) ))
          if [ $k -ge $(($row_lengh*2)) ]; then k=$(($k-$row_lengh*2)); ↩
    else k=0; fi
          flag_rollback=false
        fi
        if [ $(($iteration_count*5)) -gt $time_test ] ; then break; fi
      done
echo "    Total de iteraciones >> $( echo "$iteration_count/↩
    $no_iterations_test" | bc) "
printf "
++++++++++++++++++++++++++++++ AVE ENB csv metricstric MCS = $mcs_e ↩
    ++++++++++++++++++++++++++++++
ave_dl_brate_metric=$( echo "$ave_dl_brate_metric/$iteration_count" | bc) ↩
    \n ave_ul_brate_metric=$( echo "$ave_ul_brate_metric/$iteration_count"↩
     | bc) \n
max_ul_brate=$( echo "$max_ul_brate" | bc) \n max_dl_brate=$( echo "↩
    $max_dl_brate" | bc) \n" >> $path_to_folder/${folder_name}_ave_traces
if [ ! $max_dl_brate -eq 0 ]; then
echo -e "$mcs_e \t $max_dl_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_DL_brate_max_tracedata; fi
```

```
if [ ! $max_ul_brate -eq 0 ]; then
echo -e "$mcs_e \t $max_ul_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_UL_brate_max_tracedata; fi
fi
}

ue_csv_metrics(){
  if [ -s *_ue_metrics.csv ] ; then
  echo " >>> Scraping $(find *_ue_metrics.csv)"
  sed -i "s/;/$(printf "\t")/g" *_ue_metrics.csv
  measure_array=($( egrep '^[0-9]' *_ue_metrics.csv | cut -f-14 -d$'\t' ))
  tracedata_signal_rsrp=0 ; tracedata_signal_CarrierFrequencyOffset=0 ; ↩
    tracedata_dl_snr=0 ; tracedata_dl_brate=0 ; tracedata_dl_TimeAdvance=0↩
     ; tracedata_ul_buff=0
  tracedata_signal_pathloss=0 ; tracedata_dl_mcs=0 ; tracedata_dl_turbo=0 ↩
    ; tracedata_dl_bler=0 ; tracedata_ul_mcs=0 ; tracedata_ul_brate=0 ; ↩
    tracedata_ul_bler=0
  count=0 ; metric_time=0 ; time=0; max_ul_brate=0; max_dl_brate=0
  if [[ $iperf_mode == "server" ]]; then test_index=7 # MONITORING ↩
    DOWNLINK
  else  test_index=12 #MONITORING UPLINK
  fi
  iteration_count=0
  threshold_test=$( standardized_to_unit ${measure_array[$test_index]} )
  it_beginning_flag=-1000
  row_lengh=14
  flag_rollback=false
  flag_playtrace=false
  for ((k=0; k<${#measure_array[@]}; k++)); do
    elt=${measure_array[$k]}
    if [ $(($k%$row_lengh)) -eq $test_index ]; then threshold_test=$( ↩
    standardized_to_unit $elt ); fi
    if [ $(($iteration_count-$it_beginning_flag)) -ge $((↩
    $no_iterations_test)) ]; then flag_playtrace=false; fi
    if (( $(echo "$threshold_test >= $threshold_brate" | bc -l ) )) && [ ↩
    $flag_playtrace = "false" ] ; then
```

```
        flag_rollback=true
        flag_playtrace=true
        it_beginning_flag=$iteration_count
        sub_time_iter=0
    elif [ $flag_playtrace = "true" ]; then
        case $(($k%$row_lengh)) in
          1)
            data_signal_rsrp=$elt
            echo -e "$sub_time_iter\t $data_signal_rsrp" >> ↩
    $path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
    _csv_signal_rsrp_ue_metric
            tracedata_signal_rsrp=$( echo "$tracedata_signal_rsrp+↩
    $data_signal_rsrp" | bc -l )
            shift
            ;;
          2)
            data_signal_pathloss=${elt//%/}
            echo -e "$sub_time_iter\t $data_signal_pathloss" >> ↩
    $path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
    _csv_signal_pathloss_ue_metric
            tracedata_signal_pathloss=$( echo "$tracedata_dl_mcs+↩
    $data_signal_pathloss" | bc -l )
            shift
            ;;
          3)
            data_signal_CarrierFrequencyOffset=$( standardized_to_unit ↩
    $elt )
            echo -e "$sub_time_iter\t $data_signal_CarrierFrequencyOffset"↩
     >> $path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
    _csv_signal_CarrierFrequencyOffset_ue_metric
            tracedata_signal_CarrierFrequencyOffset=$( echo "↩
    $data_signal_CarrierFrequencyOffset+↩
    $data_signal_CarrierFrequencyOffset" | bc -l )
            shift
            ;;
          4)
```

```
        data_dl_mcs=$elt
        echo -e "$sub_time_iter\t $data_dl_mcs" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_csv_dl_mcs_ue_metric
        tracedata_dl_mcs=$( echo "$tracedata_dl_mcs+$data_dl_mcs" | bc↩
 -l )
        shift
        ;;
      5)
        data_dl_snr=$elt
        echo -e "$sub_time_iter\t $data_dl_snr" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_csv_dl_snr_ue_metric
        tracedata_dl_snr=$( echo "$tracedata_dl_snr+$data_dl_snr" | bc↩
 -l )
        shift
        ;;
      6)
        data_dl_turbo=$elt
        echo -e "$sub_time_iter\t $data_dl_turbo" >> $path_to_folder/$↩
{mcs_e}/${time_folder}/${folder_name}_csv_dl_turbo_ue_metric
        tracedata_dl_turbo=$( echo "$tracedata_dl_turbo+$data_dl_turbo↩
" | bc -l )
        shift
        ;;
      7)
        data_dl_brate=$( standardized_to_unit $elt )
        if (( $(echo "$data_dl_brate > 1" | bc -l ) )); then
          echo -e "$sub_time_iter\t $data_dl_brate" >> $path_to_folder↩
/${mcs_e}/${time_folder}/${folder_name}_csv_dl_brate_ue_metric
          tracedata_dl_brate=$( echo "$tracedata_dl_brate+↩
$data_dl_brate" | bc -l )
          if (( $(echo "$data_dl_brate > $max_dl_brate" | bc -l ) )); ↩
then max_dl_brate=$data_dl_brate; fi
        fi
        shift
        ;;
      8)
```

```
          data_dl_bler=${elt//%/}
          echo -e "$sub_time_iter\t $data_dl_bler" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_csv_dl_bler_ue_metric
          tracedata_dl_bler=$( echo "$tracedata_dl_bler+$data_dl_bler" |↩
 bc -l )
          shift
          ;;
        9)
          data_dl_TimeAdvance=$elt
          echo -e "$sub_time_iter\t $data_dl_TimeAdvance" >> ↩
$path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
_csv_dl_TimeAdvance_ue_metric
          tracedata_dl_TimeAdvance=$( echo "$tracedata_dl_TimeAdvance+↩
$data_dl_TimeAdvance" | bc -l )
          shift
          ;;
        10)
          data_ul_mcs=$elt
          echo -e "$sub_time_iter\t $data_ul_mcs" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_csv_ul_mcs_ue_metric
          tracedata_ul_mcs=$( echo "$tracedata_ul_mcs+$data_ul_mcs" | bc↩
 -l )
          shift
          ;;
        11)
          data_ul_buff=$elt
          data_ul_buff=$( standardized_to_unit $data_ul_buff )
          echo -e "$sub_time_iter\t $data_ul_buff" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_csv_ul_buff_ue_metric
          tracedata_ul_buff=$( echo "$tracedata_ul_buff+$data_ul_buff" |↩
 bc -l )
          shift
          ;;
        12)
          data_ul_brate=$( standardized_to_unit $elt )
          if (( $(echo "$data_ul_brate > 1" | bc -l ) )); then
```

```
            echo -e "$sub_time_iter\t $data_ul_brate" >> $path_to_folder/${mcs_e}/${time_folder}/${folder_name}_csv_ul_brate_ue_metric
            tracedata_ul_brate=$( echo "$tracedata_ul_brate+$data_ul_brate" | bc -l )
            if (( $(echo "$data_ul_brate > $max_ul_brate" | bc -l ) )); then max_ul_brate=$data_ul_brate; fi
            fi
            shift
            ;;
          13)
            data_ul_bler=${elt//%/}
            echo -e "$sub_time_iter\t $data_ul_bler" >> $path_to_folder/${mcs_e}/${time_folder}/${folder_name}_csv_ul_bler_ue_metric
            tracedata_ul_bler=$( echo "$tracedata_ul_bler+$data_ul_bler" | bc -l )
            shift
            ;;
        esac
      if [ $(($k%$row_lengh)) -eq 0 ]; then
        ((sub_time_iter+=5)) ; ((iteration_count++)); fi
      fi
      if [ $flag_rollback = "true" ]; then
        k=$(( $k - ($k%$row_lengh) ))
        if [ $k -ge $(($row_lengh*2)) ]; then k=$(($k-$row_lengh*2)); else k=0; fi
        flag_rollback=false
      fi
      if [ $(($iteration_count*5)) -gt $time_test ] ; then break; fi
    done
echo "    Total de iteraciones >> $( echo "$iteration_count/$no_iterations_test" | bc) "
printf "
++++++++++++++++++++++++++++++ AVE UE csv metricstric MCS = $mcs_e ++++++++++++++++++++++++++++++
tracedata_signal_rsrp=$( echo "$tracedata_signal_rsrp/$iteration_count" | bc) \n data_signal_CarrierFrequencyOffset=$( echo "
```

```
    $data_signal_CarrierFrequencyOffset/$iteration_count" | bc)
tracedata_dl_snr=$( echo "$tracedata_dl_snr/$iteration_count" | bc) \n ↩
    tracedata_dl_brate=$( echo "$tracedata_dl_brate/$iteration_count" | bc↩
    )
tracedata_dl_TimeAdvance=$( echo "$tracedata_dl_TimeAdvance/↩
    $iteration_count" | bc) \n tracedata_ul_buff=$( echo "↩
    $tracedata_ul_buff/$iteration_count" | bc)
tracedata_signal_pathloss=$( echo "$tracedata_signal_pathloss/↩
    $iteration_count" | bc) \n tracedata_dl_mcs=$( echo "$tracedata_dl_mcs↩
    /$iteration_count" | bc)
tracedata_dl_turbo=$( echo "$tracedata_dl_turbo/$iteration_count" | bc)  \↩
    n tracedata_dl_bler=$( echo "$tracedata_dl_bler/$iteration_count" | bc↩
    )
tracedata_ul_mcs=$( echo "$tracedata_ul_mcs/$iteration_count" | bc)  \n ↩
    tracedata_ul_brate=$( echo "$tracedata_ul_brate/$iteration_count" | bc↩
    )
tracedata_ul_bler=$( echo "$tracedata_ul_bler/$iteration_count" | bc) \n ↩
    max_ul_brate=$( echo "$max_ul_brate" | bc)
max_dl_brate=$( echo "$max_dl_brate" | bc) \n" >> $path_to_folder/${↩
    folder_name}_ave_traces
echo -e "$mcs_e \t $max_dl_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_DL_brate_max_tracedata
echo -e "$mcs_e \t $max_ul_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_UL_brate_max_tracedata
fi
}

srsLTE_enb_measurement(){
if [ -s autoapp_log_enb_console_* ] ; then
for log in autoapp_log_enb_console_*.log ; do
  echo " >>> Scraping $log"
  sub_console=$( echo $log | cut -d'_' -f5 | cut -d'.' -f1 )
  measure_array=($( egrep '^[0-9]' autoapp_log_enb_console_${sub_console}.↩
    log ))
  tracedata_dl_cqi=0 ; tracedata_dl_mcs=0 ; tracedata_dl_brate=0 ; ↩
    tracedata_dl_bler=0 ; tracedata_ul_snr=0
```

```
  tracedata_ul_phr=0 ; tracedata_ul_mcs=0 ; tracedata_ul_brate=0 ; ↩
    tracedata_ul_bler=0 ; tracedata_ul_bsr=0
  max_ul_brate=0; max_dl_brate=0
  if [[ $iperf_mode == "server" ]]; then test_index=9 # MONITORING ↩
    DOWNLINK
  else  test_index=4 #MONITORING UPLINK
  fi
  iteration_count=0
  threshold_test=$( standardized_to_unit ${measure_array[$test_index]} )
  it_beginning_flag=-1000
  row_lengh=12
  flag_rollback=false
  flag_playtrace=false
  for ((k=0; k<${#measure_array[@]}; k++)); do
    elt=${measure_array[$k]}
    if [ $(($k%$row_lengh)) -eq $test_index ]; then threshold_test=$( ↩
    standardized_to_unit $elt ); fi
    if [ $(($iteration_count-$it_beginning_flag)) -ge $((↩
    $no_iterations_test)) ]; then flag_playtrace=false; fi
    if (( $(echo "$threshold_test >= $threshold_brate" | bc -l ) )) && [ ↩
    $flag_playtrace = "false" ] ; then
#       echo "  iter. $k $iteration_count brate sobrepasado ↩
    $threshold_test "
        flag_rollback=true
        flag_playtrace=true
        it_beginning_flag=$iteration_count
        sub_time_iter=0
    elif [ $flag_playtrace = "true" ]; then
        case $(($k%$row_lengh)) in
          1)
            data_dl_cqi=$elt
            echo -e "$sub_time_iter \t $data_dl_cqi" >> $path_to_folder/${↩
    mcs_e}/${time_folder}/${folder_name}_tracedata_dl_cqi_enb_srsLTE
            tracedata_dl_cqi=$( echo "$tracedata_dl_cqi+$data_dl_cqi" | bc↩
     -l )
            shift
```

```
          ;;
        3)
          data_dl_mcs=$elt
          echo -e "$sub_time_iter \t $data_dl_mcs" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_tracedata_dl_mcs_enb_srsLTE
          tracedata_dl_mcs=$( echo "$tracedata_dl_mcs+$data_dl_mcs" | bc↩
 -l )
          shift
          ;;
        4)
          data_dl_brate=$( standardized_to_unit $elt )
          if (( $(echo "$data_dl_brate > 1" | bc -l ) )); then
          echo -e "$sub_time_iter \t $data_dl_brate" >> $path_to_folder/↩
${mcs_e}/${time_folder}/${folder_name}_tracedata_dl_brate_enb_srsLTE
          tracedata_dl_brate=$( echo "$tracedata_dl_brate+$data_dl_brate↩
" | bc -l )
          if (( $(echo "$data_dl_brate > $max_dl_brate" | bc -l ) )); ↩
then max_dl_brate=$data_dl_brate; fi
          fi
          shift
          ;;
        5)
          data_dl_bler=${elt//%/}
          echo -e "$sub_time_iter \t $data_dl_bler" >> $path_to_folder/$↩
{mcs_e}/${time_folder}/${folder_name}_tracedata_dl_bler_enb_srsLTE
          tracedata_dl_bler=$( echo "$tracedata_dl_bler+$data_dl_bler" |↩
 bc -l )
          shift
          ;;
        6)
          data_ul_snr=$elt
          echo -e "$sub_time_iter \t $data_ul_snr" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_tracedata_ul_snr_enb_srsLTE
          tracedata_ul_snr=$( echo "$tracedata_ul_snr+$data_ul_snr" | bc↩
 -l )
          shift
```

```
        ;;
      7)
        data_ul_phr=$elt
        echo -e "$sub_time_iter \t $data_ul_phr" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_tracedata_ul_phr_enb_srsLTE
        tracedata_ul_phr=$( echo "$tracedata_ul_phr+$data_ul_phr" | bc↩
 -l )
        shift
        ;;
      8)
        data_ul_mcs=$elt
        echo -e "$sub_time_iter \t $data_ul_mcs" >> $path_to_folder/${↩
mcs_e}/${time_folder}/${folder_name}_tracedata_ul_mcs_enb_srsLTE
        tracedata_ul_mcs=$( echo "$tracedata_ul_mcs+$data_ul_mcs" | bc↩
 -l )
        shift
        ;;
      9)
        data_ul_brate=$( standardized_to_unit $elt )
        if (( $(echo "$data_ul_brate > 1" | bc -l ) )); then
        echo -e "$sub_time_iter \t $data_ul_brate" >> $path_to_folder/↩
${mcs_e}/${time_folder}/${folder_name}_tracedata_ul_brate_enb_srsLTE
        tracedata_ul_brate=$( echo "$tracedata_ul_brate+$data_ul_brate↩
" | bc -l )
        if (( $(echo "$data_ul_brate > $max_ul_brate" | bc -l ) )); ↩
then max_ul_brate=$data_ul_brate; fi
        fi
        shift
        ;;
      10)
        data_ul_bler=${elt//%/}
        echo -e "$sub_time_iter \t $data_ul_bler" >> $path_to_folder/$↩
{mcs_e}/${time_folder}/${folder_name}_tracedata_ul_bler_enb_srsLTE
        tracedata_ul_bler=$( echo "$tracedata_ul_bler+$data_ul_bler" |↩
 bc -l )
        shift
```

```
            ;;
          11)
            data_ul_bsr=$( standardized_to_unit $elt )
            echo -e "$sub_time_iter \t $data_ul_bsr" >> $path_to_folder/${↩
    mcs_e}/${time_folder}/${folder_name}_tracedata_ul_bsr_enb_srsLTE
            tracedata_ul_bsr=$( echo "$tracedata_ul_bsr+$data_ul_bsr" | bc↩
     -l )
            shift
            ;;
          esac
          if [ $(($k%$row_lengh)) -eq 0 ]; then
            ((sub_time_iter+=5)) ; ((iteration_count++)); fi
        fi
          if [ $flag_rollback = "true" ]; then
            k=$(( $k - ($k%$row_lengh) ))
            if [ $k -ge $(($row_lengh*2)) ]; then k=$(($k-$row_lengh*2)); ↩
    else k=0; fi
            flag_rollback=false
          fi
          if [ $(($iteration_count*5)) -gt $time_test ] ; then break; fi
      done
echo "    Total de iteraciones >> $( echo "$iteration_count/↩
    $no_iterations_test" | bc) "
printf "+++++++++++++++++++++++++++++++ TRACE eNB DATA MCS = $mcs_e ↩
    +++++++++++++++++++++++++++++++
ave_data_dl_cqi=$( echo "$tracedata_dl_cqi/$iteration_count" | bc) \n ↩
    ave_data_ul_phr=$( echo "$tracedata_ul_phr/$iteration_count" | bc)
ave_data_dl_mcs=$( echo "$tracedata_dl_mcs/$iteration_count" | bc) \n ↩
    ave_data_ul_mcs=$( echo "$tracedata_ul_mcs/$iteration_count" | bc)
ave_data_dl_brate=$( echo "$tracedata_dl_brate/$iteration_count" | bc) \n ↩
    ave_data_ul_brate=$( echo "$tracedata_ul_brate/$iteration_count" | bc)
ave_data_dl_bler=$( echo "$tracedata_dl_bler/$iteration_count" | bc) \n ↩
    ave_data_ul_bler=$( echo "$tracedata_ul_bler/$iteration_count" | bc)
ave_data_ul_snr=$( echo "$tracedata_ul_snr/$iteration_count" | bc) \n ↩
    ave_data_ul_bsr=$( echo "$tracedata_ul_bsr/$iteration_count" | bc) \n ↩
    max_ul_brate=$( echo "$max_ul_brate" | bc)
```

```
max_dl_brate=$( echo "$max_dl_brate" | bc) \n" >> $path_to_folder/${↩
    folder_name}_ave_traces
printf "
++++++++++++++++++++++++++++++ AVE ICMP TIME MCS = $mcs_e Iteracion ↩
    $iteration_count ++++++++++++++++++++++++++++++
$( echo "$ave_icmp_time/$ping_count" | bc) \n" >> $path_to_folder/${↩
    folder_name}_ave_traces
echo -e "$mcs_e \t $max_dl_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_DL_brate_max_tracedata
echo -e "$mcs_e \t $max_ul_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_UL_brate_max_tracedata
done
fi
}

srsLTE_ue_measurement(){
if [ -s autoapp_log_ue_console_* ] ; then
  for log in autoapp_log_ue_console_*.log ; do
  sub_console=$( echo $log | cut -d'_' -f5 | cut -d'.' -f1 )
  measure_array=($( egrep '^ [0-9]' autoapp_log_ue_console_${sub_console}.↩
    log ))
  echo " >>> Scraping $log"
  tracedata_signal_rsrp=0 ; data_signal_CarrierFrequencyOffset=0 ; ↩
    tracedata_dl_snr=0 ; tracedata_dl_brate=0 ; tracedata_dl_TimeAdvance=0↩
     ; tracedata_ul_buff=0
  tracedata_signal_pathloss=0 ; tracedata_dl_mcs=0 ; tracedata_dl_turbo=0 ↩
    ; tracedata_dl_bler=0 ; tracedata_ul_mcs=0 ; tracedata_ul_brate=0 ; ↩
    tracedata_ul_bler=0
  max_ul_brate=0; max_dl_brate=0
  if [[ $iperf_mode == "server" ]]; then test_index=7 # MONITORING ↩
    DOWNLINK
  else  test_index=12 #MONITORING UPLINK
  fi
  iteration_count=0
  threshold_test=$( standardized_to_unit ${measure_array[$test_index]} )
  it_beginning_flag=-1000
```

```
row_lengh=14
flag_rollback=false
flag_playtrace=false
for ((k=0; k<${#measure_array[@]}; k++)); do
  elt=${measure_array[$k]}
  if [ $(($k%$row_lengh)) -eq $test_index ]; then threshold_test=$( ↩
  standardized_to_unit $elt );  fi
  if [ $(($iteration_count-$it_beginning_flag)) -ge $((↩
  $no_iterations_test)) ]; then flag_playtrace=false; fi
  if (( $(echo "$threshold_test >= $threshold_brate" | bc -l ) )) && [ ↩
  $flag_playtrace = "false" ] ; then
      flag_rollback=true
      flag_playtrace=true
      it_beginning_flag=$iteration_count
      sub_time_iter=0
  elif [ $flag_playtrace = "true" ]; then
      case $(($k%$row_lengh)) in
        1)
          data_signal_rsrp=$elt
          echo -e "$sub_time_iter \t $data_signal_rsrp " >> ↩
  $path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
  _tracedata_signal_rsrp_ue_srsLTE
          tracedata_signal_rsrp=$( echo "$tracedata_signal_rsrp+↩
  $data_signal_rsrp" | bc -l )
          shift
          ;;
        2)
          data_signal_pathloss=${elt//%/}
          echo -e "$sub_time_iter \t $data_signal_pathloss " >> ↩
  $path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
  _tracedata_signal_pathloss_ue_srsLTE
          tracedata_signal_pathloss=$( echo "$tracedata_dl_mcs+↩
  $data_signal_pathloss" | bc -l )
          shift
          ;;
        3)
```

```
        data_signal_CarrierFrequencyOffset=$( standardized_to_unit ↩
$elt )
        echo -e "$sub_time_iter \t $data_signal_CarrierFrequencyOffset↩
 " >> $path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
_tracedata_signal_pathloss_ue_srsLTE
        tracedata_signal_CarrierFrequencyOffset=$( echo "↩
$data_signal_CarrierFrequencyOffset+↩
$data_signal_CarrierFrequencyOffset" | bc -l )
        shift
        ;;
      4)
        data_dl_mcs=$elt
        echo -e "$sub_time_iter \t $data_dl_mcs " >> $path_to_folder/$↩
{mcs_e}/${time_folder}/${folder_name}_tracedata_dl_mcs_ue_srsLTE
        tracedata_dl_mcs=$( echo "$tracedata_dl_mcs+$data_dl_mcs" | bc↩
 -l )
        shift
        ;;
      5)
        data_dl_snr=$elt
        echo -e "$sub_time_iter \t $data_dl_snr " >> $path_to_folder/$↩
{mcs_e}/${time_folder}/${folder_name}_tracedata_dl_snr_ue_srsLTE
        tracedata_dl_snr=$( echo "$tracedata_dl_snr+$data_dl_snr" | bc↩
 -l )
        shift
        ;;
      6)
        data_dl_turbo=$elt
        echo -e "$sub_time_iter \t $data_dl_turbo " >> $path_to_folder↩
/${mcs_e}/${time_folder}/${folder_name}_tracedata_dl_turbo_ue_srsLTE
        tracedata_dl_turbo=$( echo "$tracedata_dl_turbo+$data_dl_turbo↩
" | bc -l )
        shift
        ;;
      7)
        data_dl_brate=$( standardized_to_unit $elt )
```

```
        if (( $(echo "$data_dl_brate > 1" | bc -l ) )); then
        echo -e "$sub_time_iter \t $data_dl_brate " >> $path_to_folder↩
/${mcs_e}/${time_folder}/${folder_name}_tracedata_dl_brate_ue_srsLTE
        tracedata_dl_brate=$( echo "$tracedata_dl_brate+$data_dl_brate↩
" | bc -l )
        if (( $(echo "$data_dl_brate > $max_dl_brate" | bc -l ) )); ↩
then max_dl_brate=$data_dl_brate; fi
        fi
        shift
        ;;
      8)
        data_dl_bler=${elt//%/}
        echo -e "$sub_time_iter \t $data_dl_bler " >> $path_to_folder/↩
${mcs_e}/${time_folder}/${folder_name}_tracedata_dl_bler_ue_srsLTE
        tracedata_dl_bler=$( echo "$tracedata_dl_bler+$data_dl_bler" |↩
 bc -l )
        shift
        ;;
      9)
        data_dl_TimeAdvance=$elt
        echo -e "$sub_time_iter \t $data_dl_TimeAdvance " >> ↩
$path_to_folder/${mcs_e}/${time_folder}/${folder_name}↩
_tracedata_dl_TimeAdvance_ue_srsLTE
        tracedata_dl_TimeAdvance=$( echo "$tracedata_dl_TimeAdvance+↩
$data_dl_TimeAdvance" | bc -l )
        shift
        ;;
      10)
        data_ul_mcs=$elt
        echo -e "$sub_time_iter \t $data_ul_mcs " >> $path_to_folder/$↩
{mcs_e}/${time_folder}/${folder_name}_tracedata_ul_mcs_ue_srsLTE
        tracedata_ul_mcs=$( echo "$tracedata_ul_mcs+$data_ul_mcs" | bc↩
 -l )
        shift
        ;;
      11)
```

```
        data_ul_buff=$( standardized_to_unit $elt )
        echo -e "$sub_time_iter \t $data_ul_buff " >> $path_to_folder/↩
${mcs_e}/${time_folder}/${folder_name}_tracedata_ul_buff_ue_srsLTE
        tracedata_ul_buff=$( echo "$tracedata_ul_buff+$data_ul_buff" |↩
 bc -l )
        shift
        ;;
      12)
        data_ul_brate=$( standardized_to_unit $elt )
        if (( $(echo "$data_ul_brate > 100" | bc -l ) )); then
        echo -e "$sub_time_iter \t $data_ul_brate " >> $path_to_folder↩
/${mcs_e}/${time_folder}/${folder_name}_tracedata_ul_brate_ue_srsLTE
        tracedata_ul_brate=$( echo "$tracedata_ul_brate+$data_ul_brate↩
" | bc -l )
        if (( $(echo "$data_ul_brate > $max_ul_brate" | bc -l ) )); ↩
then max_ul_brate=$data_ul_brate; fi
        fi
        shift
        ;;
      13)
        data_ul_bler=${elt//%/}
        echo -e "$sub_time_iter \t $data_ul_bler " >> $path_to_folder/↩
${mcs_e}/${time_folder}/${folder_name}_tracedata_ul_bler_ue_srsLTE
        tracedata_ul_bler=$( echo "$tracedata_ul_bler+$data_ul_bler" |↩
 bc -l )
        shift
        ;;
      esac
    if [ $(($k%$row_lengh)) -eq 0 ]; then
    ((sub_time_iter+=5)) ; ((iteration_count++)); fi
fi
if [ $flag_rollback = "true" ]; then
    k=$(( $k - ($k%$row_lengh) ))
    if [ $k -ge $(($row_lengh*2)) ]; then k=$(($k-$row_lengh*2)); else↩
 k=0; fi
    flag_rollback=false
```

```
    fi
    if [ $(($iteration_count*5)) -gt $time_test ] ; then break; fi
  done
echo "    Total de iteraciones >> $( echo "$iteration_count/↩
    $no_iterations_test" | bc) "
printf "+++++++++++++++++++++++++++++++ TRACE UE DATA MCS = $mcs_e ↩
    ++++++++++++++++++++++++++++++
tracedata_signal_rsrp=$( echo "$tracedata_signal_rsrp/$iteration_count" | ↩
    bc) \n data_signal_CarrierFrequencyOffset=$( echo "↩
    $data_signal_CarrierFrequencyOffset/$iteration_count" | bc)
tracedata_dl_snr=$( echo "$tracedata_dl_snr/$iteration_count" | bc) \n ↩
    tracedata_dl_brate=$( echo "$tracedata_dl_brate/$iteration_count" | bc↩
    )
tracedata_dl_TimeAdvance=$( echo "$tracedata_dl_TimeAdvance/↩
    $iteration_count" | bc) \n tracedata_ul_buff=$( echo "↩
    $tracedata_ul_buff/$iteration_count" | bc)
tracedata_signal_pathloss=$( echo "$tracedata_signal_pathloss/↩
    $iteration_count" | bc) \n tracedata_dl_mcs=$( echo "$tracedata_dl_mcs↩
    /$iteration_count" | bc)
tracedata_dl_turbo=$( echo "$tracedata_dl_turbo/$iteration_count" | bc)  \↩
    n tracedata_dl_bler=$( echo "$tracedata_dl_bler/$iteration_count" | bc↩
    )
tracedata_ul_mcs=$( echo "$tracedata_ul_mcs/$iteration_count" | bc)  \n ↩
    tracedata_ul_brate=$( echo "$tracedata_ul_brate/$iteration_count" | bc↩
    )
tracedata_ul_bler=$( echo "$tracedata_ul_bler/$iteration_count" | bc) \n ↩
    max_ul_brate=$( echo "$max_ul_brate" | bc)
max_dl_brate=$( echo "$max_dl_brate" | bc) \n" >> $path_to_folder/${↩
    folder_name}_ave_traces
printf "
++++++++++++++++++++++++++++++ AVE ICMP TIME MCS = $mcs_e ↩
    ++++++++++++++++++++++++++++++
$( echo "$ave_icmp_time/$ping_count" | bc) \n" >> $path_to_folder/${↩
    folder_name}_ave_traces
echo -e "$mcs_e \t $max_dl_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_DL_brate_max_tracedata
```

```
echo -e "$mcs_e \t $max_ul_brate \t $n_prb" >> $path_to_folder/${↩
    folder_name}_UL_brate_max_tracedata
done
fi
}

plotting(){
  declare -a arry=("${@}")
  declare -a colors=( "black" "brown" "red" "orange" "olive" "magenta" "↩
    blue" "navy blue" )
  printf "reset
  set autoscale
  set xlabel \"${arry[1]}\"
  set ylabel \"${arry[2]}\"
  set terminal png nocrop enhanced font arial 12 size 2090,1024
  #set terminal epslatex size 6.2,3.5 color colortext
  #set terminal png nocrop enhanced font arial 12 size 800,650
  set output \"${arry[3]}\"
  set grid x y back
  #set xrange [0:28]
  #set yrange [0:50]
  set key below
  " > gnuplotscript
  smooth="smooth mcsplines"; unit=""; normalize=""; mantissa=0
  echo "data1 = \"${arry[4]}\"" > gnuplotscript_aux1
  if [[ ${arry[2]} =~ "BRATE" ]];  then
  echo "set format y '%.0s%cb'" >> gnuplotscript
  unit="mbit/sec"; normalize='/1000000'; mantissa=2;
  echo -e "\n
  stats data1 using 2 nooutput name 'Y1_' \n
  stats data1 using 1 every ::Y1_index_max::Y1_index_max nooutput \n
  X1_max=STATS_max \n
  Y_min=Y1_min \n
  Y_max=Y1_max \n
  set label 1 sprintf(\"max->%.${mantissa}f $unit\", Y1_max$normalize) ↩
    right  at first X1_max,Y1_max point pt 1 ps 1 offset 0,0.5 textcolor ↩
```

```
    rgb \"${colors[0]}\" \n" >> gnuplotscript_aux1
  fi
  limit=$((${arry[0]}+1))
  printf "plot data1 using 1:2 notitle with points ps 0.5 lc rgb \"${↩
    colors[0]}\", data1 using 1:2 $smooth title \"${arry[5]}\" with lines ↩
    lc rgb \"${colors[0]}\" " > gnuplotscript_aux2
  for i in $( seq 3 $limit ); do
    echo "data$(($i-1)) = \"${arry[$i*2]}\"" >> gnuplotscript_aux1
    if [[ ${arry[2]} =~ "BRATE" ]];  then
    echo -e "\n
    stats data$(($i-1)) using 2 nooutput name 'Y$(($i-1))_' \n
    stats data$(($i-1)) using 1 every ::Y$(($i-1))_index_max::Y$(($i-1))↩
    _index_max nooutput \n
    X$(($i-1))_max = STATS_max \n
    if (Y_min > Y$(($i-1))_min) Y_min=Y$(($i-1))_min \n
    if (Y_max < Y$(($i-1))_max) Y_max=Y$(($i-1))_max \n
    set label $(($i-1)) sprintf(\"max->%.${mantissa}f $unit\", Y$(($i-1))↩
    _max$normalize) center at first X$(($i-1))_max,Y$(($i-1))_max point pt↩
     $(($i-1)) ps 1 offset 0,0.5 textcolor rgb \"${colors[($i-2)%8]}\" \n"↩
     >> gnuplotscript_aux1
    fi
    printf ", data$(($i-1)) using 1:2 notitle with points ps 0.5 lc rgb\"$↩
    {colors[($i-2)%8]}\" , data$(($i-1)) using 1:2 $smooth title \"${arry[↩
    $i*2+1]}\" with lines lc rgb \"${colors[($i-2)%8]}\" " >> ↩
    gnuplotscript_aux2
  done
  if [[ ${arry[2]} =~ "BRATE" ]];  then echo -e "if (Y_max > 0) { ↩
    Y_max_limit=Y_max+Y_max*0.05 \n set yrange [Y_min:Y_max_limit] }" >> ↩
    gnuplotscript_aux1; fi
  cat gnuplotscript_aux1 >> gnuplotscript
  cat gnuplotscript_aux2 >> gnuplotscript
  rm gnuplotscript_aux1 gnuplotscript_aux2
  gnuplot gnuplotscript
}

plott_data(){
```

```
rm ${path_to_folder}toplot/GENERAL_*.dat && echo "cleaning"
### PLOTTING
plot_data_type=( $( cat ~/trace_elements ) )
cd ${path_to_folder}toplot/
for trace in "${plot_data_type[@]}"; do
  first=0
  plotting_count=0
  declare -a toPlot=()
  if [ -s *_${trace} ] && ([ $trace = "icmp" ] || [ $trace = "↩
  adapterstatistics" ] || [ $trace = "DL_brate_max_tracedata" ] || [ ↩
  $trace = "UL_brate_max_tracedata" ]); then
      echo "...Ploting $trace..."
      f=$( find *_${trace})
      title=$(echo ${f^^} | cut -d'_' -f5)
      no_prb=("6" "15" "25" "50" "75" "100")
      channel=${root_folder##*/}
      plotting_count=0
      toPlot=( "MCS" "$title" "${path_to_folder}toplot/$title_$trace.png↩
  " )
      location_temp=$(pwd)
      cd .. ; cd .. ; cd ..
      root_location=$(pwd)
      for prb in "${no_prb[@]}"; do
        if [ -s ${prb}* ]; then
        cd $prb* ; cd $channel ; cd *toplot
        if [ -s *_${trace} ]; then
          temp_path_to_folder=$(pwd)
          f=$( find *_${trace})
          sub_plot=$(echo ${f^^} | cut -d'_' -f2)
          f=$temp_path_to_folder/$f
          toPlot+=( "$f" "$sub_plot" )
          ((plotting_count++))
          if [ $trace = "adapterstatistics" ]; then
            cat ${f} >> ${path_to_folder}toplot/GENERAL_${channel}_${prb↩
  }_adapterstatistics.dat
          elif [ $trace = "DL_brate_max_tracedata" ]; then
```

```
            cat ${f} >> ${path_to_folder}toplot/GENERAL_${channel}_${prb}_DL_brate_max_tracedata.dat
          elif [ $trace = "UL_brate_max_tracedata" ]; then
            cat ${f} >> ${path_to_folder}toplot/GENERAL_${channel}_${prb}_UL_brate_max_tracedata.dat
          fi
        fi
        cd $root_location
        fi
      done
      cd $location_temp
  else
    for mcs_e in "${mcs_order[@]}" ; do
    if [ -s $mcs_e ]; then
      cd $mcs_e
      for time_folder in * ; do
        cd $time_folder
        if [ -s *_${trace} ]; then
          plot_count=$( find *_${trace} | wc -l )
          if [[ $plot_count -gt 0 ]] ; then
            plotting_count=$(($plotting_count+$plot_count))
            echo "...Ploting $trace... for MCS=$mcs_e..."
            for f in *_${trace} ; do
              output_folder=$(echo $f | cut -d'_' -f7)  ${ans^}
              title="${output_folder^^} $(echo ${f^^} | cut -d'_' -f6) "
              sub_plot="mcs=$mcs_e:$time_folder"
              f=${path_to_folder}toplot/$mcs_e/$time_folder/$f
              if [ $first -eq 0 ]; then mkdir -p ${path_to_folder}toplot/$output_folder; toPlot=( "TIEMPO" "$title" "${path_to_folder}toplot/$output_folder/$title_$trace.png" ); first=1; fi
              toPlot+=( "$f" )
              toPlot+=( "$sub_plot" )
            done
          fi
        fi
        cd ..
```

```
        done
        cd ..
        fi
    done
    fi
    if [[ $plotting_count -gt 0 ]] ; then
      toPlot=("$plotting_count" "${toPlot[@]}")
      plotting "${toPlot[@]}"
    fi
  done
}

############## VARS  ##############
root_folder=$(pwd)
folder_name="${root_folder##*/}"_"$(echo $root_folder | cut -d'/' -f7)"
path_to_folder=$root_folder/${folder_name}_DATA
t_mode=1
time_test=700
tx_gain=*
mcs_order=( "0" "4" "8" "12" "16" "20" "24" "25" "26" "27" "28" )
ue_category=*
no_hilos=20
no_iterations=5
iperf_mode=*
n_prb=*
srslte_mode=*
measure_mode=*active
threshold_brate=100000
no_iterations_test=27
declare -a iterations_register

scraping_data(){
  rm -r -f $path_to_folder
  mkdir $path_to_folder
  for mcs_e in "${mcs_order[@]}"; do
    if [ -s $mcs_e ]; then
```

```
      cd $mcs_e
      echo "MCS >> $mcs_e"
      for time_folder in * ; do
        cd $time_folder
        for f in * ; do
          srslte_mode=$(echo $f | cut -d'.' -f2)
          if [ -n "${srslte_mode}" ] && [ ! "${srslte_mode}" == "*" ] ; ↩
    then
            echo "...Entrando a $f..."
            cd $f; mcs=$(echo $f | cut -d'.' -f4)
            iperf_mode=$(echo $f | cut -d'.' -f8)
            n_prb=$(echo $f | cut -d'.' -f3)
            ##### adapterstatistics MEASUREMENT
            mkdir -p $path_to_folder/${mcs_e}/${time_folder}/
            adapterstatistics_measurement
            ##### srsLTE TRACE MEASUREMENT
            if [[ -n $scan_mode ]] && [ $scan_mode = "all" ]; then
              if [ $srslte_mode = "ue" ]; then srsLTE_ue_measurement ↩
    $mcs_e ;
              ue_csv_metrics $mcs_e
            else srsLTE_enb_measurement $mcs_e ;
                enb_csv_metrics $mcs_e ;
              fi
            fi
            ##### ping MEASUREMENT
            ping_measurement
            cd ..
          fi
        done
        cd ..
    done
      cd ..
    fi
  done
}
################## MAIN ##################
```

```
scan_mode=$1
if [[ -n $scan_mode ]]; then
scraping_data; fi
plott_data "${mcs_order[@]}"
```

Acrónimos y abreviaturas

3GPP	Third Generation Partnership Project.
ADC	Analog-to-Digital Converter.
AES	Advanced Encryption Standard.
AGC	Automatic Gain Control.
AM	Acknowledged Mode.
APN	Access Point Name.
ARQ	Automatic Repeat Request.
BLER	Block Error Rate.
CDF	Cumulative Distribution Function.
CFO	Carrier Frequency Offset.
CP	Cyclic Prefix.
CQI	Channel Quality Indicator.
CSI	Channel State Information.
DHCP	Dynamic Host Configuration Protocol.
DNS	Domain Name Server.
DSP	Digital Signal Processor.
E-UTRAN	Evolved UTRAN.
EARFCN	Evolved Absolute Radio Frequency Channel Number.
ECGI	E-UTRAN Cell Global Identifier.
ECI	E-UTRAN Cell Identity.
ECM	EPS Connection Management.
EMM	EPS Mobility Management.
EPC	Evolved Packet Core.
EPS	Evolved Packet System.

EPS-AKA	Evolved Packet System-Authentication and Key Agreement.
FDD	Frequency División Duplexing.
FPGA	Field Programmable Gate Array.
GPP	General Purpose Processors.
GPRS	General Packet Radio Service.
GPSDO	GPS Disciplined Oscillator.
GSM	Global System for Mobile.
GTP-C	GPRS Tunnelling Protocol - Control Plane.
GTP-U	GPRS Tunneling Protocol – User Plane.
GUMMEI	Globally Unique Mobility Management Entity Identifier.
GUTI	Globally Unique Temporary Identity.
HARQ	Hybrid Automatic Repeat Request.
HSPA+	High Speed Packet Access plus.
HSS	Home Subscriber Server.
ICMP	Internet Control Message Protocol.
IMEI	International Mobile Equipment Identity.
IMSI	International Mobile Subscriber Identity.
IMT-Advanced	International Mobile Telecommunications-Advanced.
IoT	Internet of Things.
IP	Internet Protocol.
LTE	Long Term Evolution.
MAC	Medium Access Control.
MBMS	Multimedia Broadcast Multicast Services.
MCC	Mobile Country Code.
MCS	Modulation and Coding Scheme.
ME	Mobile Equipment.
MIB	Master Information Block.
MIMO	Multiple-Input Multiple-Output.
MME	Mobility Management Entity.
MMEC	MME Code.
MMEGI	MME Group Identity.
MNC	Mobile Network Code.
NAS	Non-Access Stratum.

OAI	OpenAirInterface.
OFDM	Orthogonal Frequency Division Multiplex.
OFDMA	Orthogonal Frequency Division Multiple Access.
OSA	OpenAirInterface Software Alliance.
OTA	Over-the-Air.
PAPR	Peak to Average Power Ratio.
PCI	Physical Cell Identity.
PCRF	Policy and Charging Rules Function.
PDCP	Packet Data Convergence Protocol.
PDN	Packet Data Network.
PDSCH	Physical Uplink Shared Channel.
PDU	Protocol Data Unit.
PGW	Packet Data Network Gateway.
PLMN-ID	Public Land Mobile Network Identity.
PMI	Precoding Matrix Indicator.
PMIPv6	Proxy Mobile IPv6.
PRB	Physical Resource Blocks.
PSS	Primary Synchronization Signal.
QoS	Quality of Service.
RA-RNTI	Random Access Radio Network Temporary Identity.
RE	Resource Element.
RI	Rank Indicator.
RLC	Radio Link Control.
ROHC	Robust Header Compression.
RRC	Radio Resource Control.
RSRP	Reference Signals Received Power.
RTT	Round-Trip Delay Time.
S1-APP	S1 - Application Part.
SAE	System Architecture Evolution.
SC-FDMA	Single Carrier Frequency Division Multiple Access.
SCTP	Stream Control Transmission Protocol.
SDN	Software Defined Networking.
SDR	Software Defined Radio.
SGW	Serving Gateway.
SIB	System Information Block.

SINR	Signal to Interference plus Noise Ratio.
SNR	Signal to Noise Ratio.
SRB	Signalling Radio Bearer.
SRS	Software Radio Systems Company.
SSS	Secondary Synchronization Signal.
TA	Tracking Areas.
TAC	Tracking Area Code.
TAI	Tracking Area Identity.
TAL	Tracking Area list.
TAM	Tracking Area Management.
TBS	Transport Block Size.
TCP	Transmission Control Protocol.
TDD	Time Division Duplex.
TEID	Tunnel End Point IDentifier.
TFT	Traffic Flow Template.
TM	Transparent Mode.
TMSI	Temporary Mobile Subscriber Identity.
TS	Technical specification.
TTI	Transmission Time Interval.
UDP	User Datagram Protocol.
UE	User Equipment.
UHD	USRP Hardware Driver.
UICC	Universal Integrated Circuit Card.
UM	Unacknowledged Mode.
USIM	Universal SIM.
USRP	Universal Software Radio Peripheral.
VoIP	Voice over IP.
WCDMA	Wideband Code Division Multiple Access.

Bibliografía

[1] "Cisco visual networking index: Forecast and trends, 2017–2022 white paper." 1, 2

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5g Be?," *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 1065–1082, June 2014. 1

[3] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-Defined and Virtualized Future Mobile and Wireless Networks: A Survey," *Mobile Networks and Applications*, vol. 20, pp. 4–18, Feb. 2015. arXiv: 1409.0079. 3

[4] "Open source SDR LTE software suite from Software Radio Systems (SRS): srsLTE/srsLTE," Mar. 2019. original-date: 2013-12-06T13:53:04Z. 3, 50, 56

[5] "SRS | Software Radio Systems." 3

[6] "USRP Hardware Driver (UHD) - Ettus Research." 3, 49

[7] C. Preimesberge, "Cisco report: Mobile phones will number 5.5 billion by 2021.," *eWeek*, p. 1, 2017. 7

[8] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013. 8, 19, 33, 36

[9] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*. Wiley, 2011. 9

[10] M. Koziol, "5g New Radio and What Comes Next," Jan. 2018. 9

[11] S. C. R. Definitions, "Sdrf-06-r-0011-v1. 0.0, 8 nov 2007," *URL (2014-10-13): http://groups. winnforum. org/d/do/1585*. 9

[12] R. Comes and F. Vodafone, *LTE: nuevas tendencias en comunicaciones móviles*. Fundación Vodafone España, 2010. 10, 18, 28, 31, 86

[13] "openLTE." 10

[14] "Arquitectura para Aplicaciones LTE de LabVIEW Communications - National Instruments." 11

[15] "OpenAirInterface – 5g software alliance for democratising wireless innovation." 11

[16] "Acquiring Our Platforms | Open Air Interface." 11

[17] N. Nikaein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab, "Demo: Openairinterface: An open lte network in a pc," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, (New York, NY, USA), pp. 305–308, ACM, 2014. 11

[18] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: An Open-Source Platform for LTE Evolution and Experimentation," *arXiv:1602.04629 [cs]*, Feb. 2016. arXiv: 1602.04629. 11, 12, 51

[19] "SRS | Software Radio Systems." 11

[20] Z. Geng, X. Wei, H. Liu, R. Xu, and K. Zheng, "Performance analysis and comparison of gpp-based sdr systems," in *2017 7th IEEE International Symposium on Microwave, Antenna, Propagation, and EMC Technologies (MAPE)*, pp. 124–129, IEEE, 2017. 12, 44

[21] C. Andrés Ramiro, C. Fiandrino, A. Blanco Pizarro, P. Jiménez Mateo, N. Ludant, and J. Widmer, "OpenLEON: An End-to-End Emulator from the Edge Data Center to the Mobile Users," in *Proceedings of the 12th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, WiNTECH '18, (New York, NY, USA), pp. 19–27, ACM, 2018. event-place: New Delhi, India. 12, 13

[22] T. M. Kavyashree, L. S. Jyothi, and U. Shetty, "Design and Implementation of 4G LTE Components -eNodeB and UE on SDR platform using srsLTE," *ITSI Transactions on Electrical and Electronics Engineering*, 2016. 13

[23] M. R. Sama, S. B. H. Said, K. Guillouard, and L. Suciu, "Enabling network programmability in lte/epc architecture using openflow," in *2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 389–396, IEEE, 2014. 16, 27

[24] M. Olsson, S. Rommer, C. Mulligan, S. Sultana, and L. Frid, *SAE and the Evolved Packet Core: Driving the mobile broadband revolution*. Academic Press, 2009. 16

[25] "3gpp ts 27.007, "at command set for user equipment (ue)"." 21

[26] 3GPP, "Evolved universal terrestrial radio access (e-utra); user equipment (ue) radio access capabilities," *TS 36.306 v10.15.0 Release 10*, 2016. 21

[27] 3GPP, "Numbering, addressing and identification; release 11, sections 2, 6, 19," *TS 23.003*, 2013. 22

[28] C. Cox, *An introduction to LTE: LTE, LTE-advanced, SAE and 4G mobile communications*. John Wiley & Sons, 2012. 22, 23

[29] 3GPP, "Mobile radio interface signalling layer 3; general aspects, release 11, section 11.2.3.1.5," *TS 24.007*, 2012. 24

[30] H. Holma and A. Toskala, *LTE for UMTS: OFDMA and SC-FDMA based radio access*. John Wiley & Sons, 2009. 26, 32

[31] 3GPP, "Evolved universal terrestrial radio access (e-utra); physical channels and modulation," *3GPP TS 36.211 V10.7.0*, 2013. 25

[32] A. N. Bikos and N. Sklavos, "Lte/sae security issues on 4g wireless networks," *IEEE Security & Privacy*, vol. 11, no. 2, pp. 55–62, 2012. 26

[33] 3GPP, "Evolved universal terrestrial radio access (e-utra); radio resource control (rrc); protocol specification," *TS 36.331*, 2011. 29

[34] 3GPP, "Non-access-stratum (nas) protocol for evolved packet system (eps); stage 3," *TS 24.301*, 2011. 29

[35] C. Cox, *An Introduction to LTE: LTE, LTE-Advanced, SAE and 4G Mobile Communications*. Wiley, 2012. 30, 38

[36] 3GPP, "Evolved universal terrestrial radio access network (e-utran); s1 application protocol (s1ap)," *TS 36.413*, p. V10, 2011. 31

[37] R. Stewart, Q. Xie, K. Morneault, *et al.*, "Ietf rfc 4960, stream control transmission protocol," *Online verfügbar unter: https://tools. ietf. org/html/rfc4960, letzter Zugriff am*, vol. 1, p. 2018, 2007. 31

[38] M. Olsson, C. Mulligan, S. Sultana, S. Rommer, and L. Frid, *EPC and 4G packet networks: driving the mobile broadband revolution*. Academic Press, 2012. 31

[39] 3GPP, "Evolved general packet radio service (gprs) tunnelling protocol for control plane (gtpv2-c); stage 3," *TS 29.274*, 2013. 31

[40] A. Larmo, M. Lindström, M. Meyer, G. Pelletier, J. Torsner, and H. Wiemann, "The lte link-layer design," *IEEE Communications magazine*, vol. 47, no. 4, pp. 52–59, 2009. 35

[41] 3GPP, "Evolved universal terrestrial radio access (e-utra); user equipment (ue) radio transmission and reception," *TS 36.101 V10.28.0*, 2018. 37

[42] 3GPP, "Evolved universal terrestrial radio access (e-utra); physical layer procedures," *TS 36.213 V 10.13.0*, 2015. 39, 40, 72

[43] S. Sesia, I. Toufik, and M. Baker, *LTE-the UMTS long term evolution: from theory to practice*. John Wiley & Sons, 2011. 39

[44] B. Bing, *Broadband wireless multimedia networks*, vol. 99. John Wiley & Sons, 2012. 41

[45] S. Sesia, I. Toufik, and M. Baker, *LTE-the UMTS long term evolution: from theory to practice*. John Wiley & Sons, 2011. 42

[46] E. U. T. R. Access, "Base station (bs) radio transmission and reception, 3gpp ts 36.104," *V14*, vol. 3, 2009. 42

[47] E. U. T. R. Access, "User equipment (ue) radio transmission and reception (release 10); 3gpp ts 36.101," *V10*, vol. 3, 2011. 42

[48] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker, "Sora: high-performance software radio using general-purpose multi-core processors," *Communications of the ACM*, vol. 54, no. 1, pp. 99–107, 2011. 43

[49] M. N. Sadiku and C. M. Akujuobi, "Software-defined radio: a brief overview," *Ieee Potentials*, vol. 23, no. 4, pp. 14–15, 2004. 43

[50] M. Dillinger, K. Madani, and N. Alonistioti, *Software defined radio: Architectures, systems and functions*. John Wiley & Sons, 2005. 44

[51] "Software Defined Radio: Past, Present, and Future - National Instruments." 44

[52] S. Sun, M. Kadoch, L. Gong, and B. Rong, "Integrating network function virtualization with SDR and SDN for 4g/5g networks," *IEEE Network*, vol. 29, pp. 54–59, May 2015. 44

[53] M. Palkovic, P. Raghavan, M. Li, A. Dejonghe, L. Van der Perre, and F. Catthoor, "Future software-defined radio platforms and mapping flows," *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 22–33, 2010. 44

[54] B. Riyanto, A. Z. Langi, A. Kurniawan, E. Marpanaji, A. Mahendra, and T. Liung, "Software architecture of software-defined radio (sdr)," 45

[55] M. Ettus *et al.*, "Ettus research, llc," *Online information on USRP board). http://www. ettus. com*, 2008. 44

[56] M. Dardaillon, K. Marquet, T. Risset, and A. Scherrer, "Software defined radio architecture survey for cognitive testbeds," in *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 189–194, IEEE, 2012. 45

[57] T. Schmid, O. Sekkat, and M. B. Srivastava, "An experimental study of network performance impact of increased latency in software defined radios," in *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pp. 59–66, ACM, 2007. 45, 86

[58] K. Vachhani and R. A. Mallari, "Experimental study on wide band fm receiver using gnuradio and rtl-sdr," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1810–1814, IEEE, 2015. 46, 50

[59] D. Valerio, "Open source software-defined radio: A survey on gnuradio and its applications," *Forschungszentrum Telekommunikation Wien, Vienna, Technical Report FTW-TR-2008-002*, 2008. 47

[60] F. A. Hamza, "The usrp under 1.5 x magnifying lens!," *Online] https://microembedded. googlecode. com/files/USRP_Documentation. pdf*, 2008. 46

[61] E. R. Brand, a National Instruments, "USRP B210 USB Software Defined Radio (SDR)." 47

[62] E. R. Brand, a National Instruments, "USRP X310 High Performance Software Defined Radio." 48

[63] "GNU Radio - The Free & Open Source Radio Ecosystem · GNU Radio." 49

[64] R. . S. International, "R&S®FSH handheld spectrum analyzer." 63

[65] ITU-R, "Requirements related to technical performance for imt-advanced radio interface (s)," *International Telecommunications Union*, 2008. 85