UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Configuración y pruebas de rendimiento de tecnologías móviles 4G y 5G

Departamento de Teoría de la Señal, Telemática y Comunicaciones

Autor
Javier Jiménez González

Directores
Jorge Navarro Ortiz y Félix Delgado Ferro

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Julio de 2023

Configuración y pruebas de rendimiento de tecnologías móviles 4G y 5G

Departamento de Teoría de la Señal, Telemática y Comunicaciones

Autor
Javier Jiménez González

Directores
Jorge Navarro Ortiz y Félix Delgado Ferro

Configuración y pruebas de rendimiento de tecnologías móviles 4G y 5G

Javier Jiménez González

Palabras clave: 4G, 5G-NSA, 5G-SA, LTE-M, NB-IoT, rendimiento y latencia

Resumen

El despliegue y la configuración de las tecnologías móviles de 4G y 5G, es un tema de gran importancia en la actualidad, debido a la gran demanda tanto de velocidad como de capacidad. Por ello, en este proyecto se pretende configurar y probar diferentes tecnologías como son LTE, LTE-M, NB-IoT, 5G NSA, 5G-SA y analizar los datos conseguidos para concluir que tecnologías obtienen mejores resultados dependiendo de distintas variables que irán cambiando en cada prueba.

En la primera parte de nuestra memoria, se contextualiza 4G y 5G para tener una idea de cómo se encuentran estas tecnologías móviles en la actualidad y como se desarrollarán en el futuro. Se expone la evolución prevista junto a estimaciones sobre el consumo de datos para los próximos años. Se describe el servicio de IoT y como día a día se sigue expandiendo más, tanto en hogares como en industria. Esta sección incluye las motivaciones tanto personales como de interés académico por las que se decidió realizar este proyecto junto a un resumen de la estructura completa de la memoria.

En la sección de estado del arte, se estudian las distintas alternativas que hay en el mercado para la implementación y despliegue de 4G y 5G. Finalmente, tras comparar todas las opciones se eligió el software de Amarisoft por las funcionalidades que nos ofrece. En el tercer apartado, se exponen los fundamentos teóricos de 4G, 5G, LTE-M y NB-IoT para conseguir un contexto sobre las tecnologías con las que se van a trabajar. Se abordan sus características principales, su arquitectura, pila de protocolos y el espectro y frecuencias utilizadas en España.

Siguiendo con el cuarto punto, se detallan los costes del proyecto, se dividen en hardware, software y humanos. Para los componentes físicos se utiliza una de las estaciones base disponibles por parte del tutor (estación base Amarisoft) y modem Quectel RM500Q-GL como dispositivo móvil para LTE, 5G NSA y 5G-SA, así como las motas FiPy de Pycom para LTE-M y NB-IoT. Del mismo modo se indica el tiempo que ha tomado el proyecto en una planificación temporal. Para la parte de configuración y pruebas se

explica cómo se debe configurar el software de Amarisoft junto a los archivos de configuración y a los dispositivos nombrados previamente. Se describen las pruebas realizadas para cada una de las tecnologías y también se exponen las variables que se cambiarán para la ejecución de las pruebas. El objetivo principal será configurar una red 4G, 5G y realizar pruebas de conectividad desde los modems, incluyendo pruebas de rendimiento y latencia. Como objetivo secundario se realizará la configuración y prueba de conectividad de una estación base a una mota NB-IoT junto a otra LTE-M en el mismo instante de tiempo.

Por último, las secciones de análisis de resultados y conclusiones y líneas futuras. En el primero se comparan los datos obtenidos entre las diferentes tecnologías mediante gráficas CDF. En el último apartado, se concluye con que redes se obtiene un mejor resultado y cuáles han sido las reflexiones finales. Además, se añade un subapartado de líneas futuras para indicar las perspectivas de mejora que tiene el proyecto.

Project Title

Javier Jiménez González

Keywords: 4G, 5G-NSA, 5G-SA, LTE-M, NB-IoT, throughput and latency

Abstract

The deployment and configuration of 4G and 5G mobile technologies is an important topic nowadays, due to the high demand for speed and capacity. For this reason, this project aims to configure and test different technologies such as LTE, LTE-M, NB-IoT, 5G NSA, 5G-SA and analyse the data obtained to conclude which technologies obtain better results depending on the different variables that will be changed.

In the first part of our paper, we contextualise 4G and 5G to get an idea of where these mobile technologies are today and how they will develop in the future. It sets out the expected developments along with estimates of data consumption for the next few years. It describes the IoT service and how it continues to expand more and more every day, both in homes and in industry. This section includes both the personal and academic motivations for the decision to do this project along a summary of the full structure of the report.

In the state of the art section, the different alternatives available on the market for the implementation and deployment of 4G and 5G are studied. Finally, after a comparison of all the options, the Amarisoft software was chosen for the functionalities it offers. In the third section, the theoretical foundations of 4G, 5G, LTE-M and NB-IoT are presented in order to provide a context about the technologies we are going to work with. Their main characteristics, architecture, protocol stack and the spectrum and frequencies used in Spain are discussed.

Following the fourth section, the costs of the project are detailed, divided into hardware, software and human resources. For the physical components, one of the base stations available from the tutor (Amarisoft base station) and Quectel RM500Q-GL modem is used as a mobile device for LTE, 5G NSA and 5G-SA, as well as Pycom's FiPy motes for LTE-M and NB-IoT. The time taken for the project is also indicated in a time schedule. For the configuration and testing part of the fifth section, it is explained how the Amarisoft software has to be configured together with the configuration files and the previously named devices. The variables to be changed for the execution of the tests are described. The tests performed for each of the

technologies are also described in this section. The main objective will be to configure a 4G, 5G network and perform connectivity tests from the modems, including throughput and latency tests. A secondary objective will be to configure and test connectivity from a base station to an NB-IoT Pycom next to an LTE-M Pycom at the same instant of time.

Finally, analysis of results and conclusions and future lines sections. The first section compares the data obtained between the different technologies using CDF graphs. The last section concludes with the networks with the best results and the final obtains. In addition, a sub-section of future lines is added to indicate the prospects for improving the project.

Yo, **Javier Jiménez González**, alumno de la titulación GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77691040F, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Javier Jiménez González

Granada a 13 de Julio de 2023.

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

D. **Félix Delgado Ferro**, alumno de doctorado del Área de Ingeniería Telemática del Departamento Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Configuración y pruebas de rendimiento de tecnologías móviles 4G y 5G***, ha sido realizado bajo su supervisión por **Javier Jiménez González**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 13 de Julio de 2023.

El director:

Jorge Navarro Ortiz **Félix Delgado Ferro**

Agradecimientos

Quiero expresar mis agradecimientos a todas las personas que han estado conmigo a lo largo de estos 4 años que han sido tan duros pero fugaces, por apoyarme y siempre contar con su ayuda.

En primer lugar, agradecer a Jorge y Félix, por guiarme y brindarme su ayuda siempre que la he necesitado en el transcurso de este proyecto.

A mi familia, que siempre me han apoyado y han creído en mi. Especialmente en los momentos más duros, como en la pandemia donde fueron pilares fundamentales para que fuese capaz de llegar hasta aquí.

A mis amigos, que muchos de ellos he conocido durante estos años en el grado de Ingeniería de Telecomunicaciones. Por todas esas tardes de estudio pero sobre todo de risas y de momentos inolvidables que han dejado una huella única en mi vida universitaria.

Por último, agradecer a Marina. La persona que durante este último año ha aguantado todos mis momentos de agobio y me ha estado animando tanto en las malas como en las buenas. Has sido fundamental para mí.

Muchas gracias de nuevo a todas las personas mencionadas anteriormente y a aquellas que me han influenciado a lo largo de mi trayectoria académica.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto y motivación

En la última década se ha producido una explosión en las tecnologías de comunicaciones móviles, lo que ha llevado, a una constante evolución debido a la creciente necesidad de los usuarios de estar conectados y enviar grandes cantidades de datos a velocidades cada vez más altas. Se estima que la cantidad consumida de datos a nivel mundial en 2022 fue de 97 zettabytes y aumentará a 181 zettabytes para 2025 [1].

Actualmente, la Cuarta Generación de comunicaciones móviles (4G) se encuentra totalmente implantada aunque comienza a encontrarse obsoleta debido a la gran demanda. Por tanto, las tecnologías de comunicaciones móviles empiezan a converger en la Quinta Generación (5G) extendiendose rápidamente por todo el planeta. Según la *Global Mobile Suppliers Association* (GSA), en junio de 2022 el 5G se encontraba de forma comercial en unos 70 países [2] y para finales de 2022 había 1150 millones de suscriptores en 5G, lo que supuso un crecimiento del 85,9 % con respecto al 2021. Se prevé un crecimiento de hasta 4.2 billones de usuarios de 5G para 2026 [3], como se muestra en la Figura 1.1.

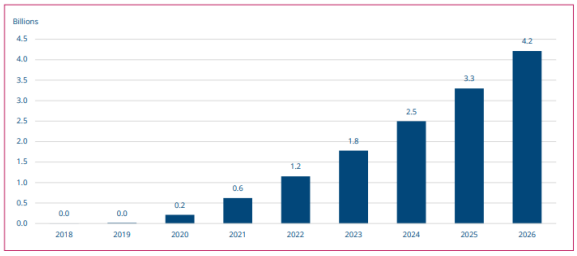

Figura 1.1: Previsión usuarios 5G [3]

Actualmente se encuentra en desarrollo la tecnología de 5G advanced a través del *Release 18* [4] de 3GPP (*Third Generation Partnership Project*). Esto indica que 5G terminó su desarrollo en 2022 y en estos momentos se está estudiando una versión mejorada que se extenderá hasta 2024 como se puede ver en la Figura 1.2. Es importante destacar que el 5G advanced ofrecerá un mayor rendimiento en términos de velocidad y capacidad de conexión, lo que mejorará la experiencia del usuario y permitirá el desarrollo de nuevas aplicaciones y servicios.

El emergente IoT (*Internet of Things*) ha estado evolucionando desde inicios de 1990 y cada vez tiene más importancia en nuestro día a día. Se tienen más aparatos electrónicos con conectividad a internet, así como bombillas inteligentes, todo tipo de sensores, electrodomésticos inteligentes entre otros. Pero no solo ocurre en los casos domésticos sino también en la industria, donde IoT tiene un impacto directo en la industria 4.0. La digitalización, los mantenimientos predecibles o la automatización autoorganizada son indispensables para impulsar tanto fábricas como ciudades inteligentes. Todo esto, se conecta a través de una red, en la mayoría de los casos Wi-Fi (*Wireless Fidelity*), por tanto, las empresas sugirieron NB-IoT (*Narrowband IoT*) que se estandarizó en 2016 en el *Release 13* [5], dando cobertura a miles de dispositivos en una sola celda [6].

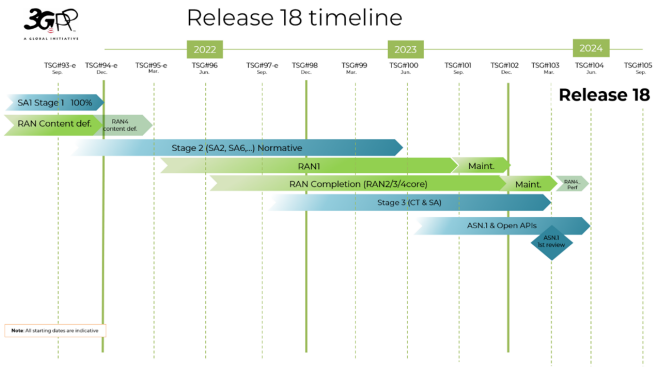

Figura 1.2: Previsión desarrollo *Release 18* [4]

Por ello, una de las motivaciones del proyecto consistirá en la implementación de las tecnologías 4G y 5G, pero centrándose en esta última. El 5G se encuentran en pleno desarrollo y con un futuro muy prometedor. Además, su despliegue se está realizando globalmente y de una manera rápida. Por tanto, es de interés conocer las diferentes alternativas que existen actualmente para desplegar estas redes. Así como el futuro de IoT en la industria y en nuestra vida cotidiana y cómo cada año contamos con más dispositivos que se conectan a la red a través de estas tecnologías de baja potencia.

Otra de las motivaciones, pero en este caso personal, es el interés que me provocaron las tecnologías móviles y la diferencia que hay entre el 5G y su predecesor en cuanto a capacidades. Consiguiendo con este proyecto, comparar ambas tecnologías y poder desplegarlas de forma experimental. Además, de la curiosidad que me despertó las IoT ya que era una tecnología que había empezado a sonar bastante en la última década.

1.2. Objetivos del proyecto

Los objetivos de este proyecto consisten en:

- **Desplegar y configurar una red 4G:** Se debe configurar una red LTE para realizar pruebas de conectividad para obtener y analizar sus resultados. Posteriormente, comparar los datos de *throughput* y latencia y observar con que parámetros se obtiene un mejor resultado.
- **Desplegar y configurar una red 5G:** En este caso se debe configurar dos tipos de redes para esta tecnología móvil, 5G-NSA y 5G-SA. Se van a realizar pruebas de conectividad y analizar los resultados del *throughput* y latencia para compararlos con las demás tecnologías.
- **Desplegar y configurar una red LTE-M:** Para esta red de IoT únicamente se realizarán pruebas de conectividad y un análisis de sus mensajes.
- **Desplegar y configurar una red NB-IoT:** Al igual que para LTE-M, se debe configurar la red para esta tecnología y realizar pruebas de conectividad y analizar los mensajes enviados.

1.3. Estructura de la memoria

En esta sección, se identifican los distintos apartados en los que se divide este proyecto, junto con una breve descripción de cada uno:

- **Capítulo 1. Introducción:** Se trata del primer capítulo de la memoria, donde se realiza un visión general del proyecto, se compone de las siguientes partes:
  - **Contexto y motivación:** Partimos del contexto del problema a resolver y cual fue la motivación para elegirlo.
  - **Objetivos del proyecto:** Se exponen los objetivos principales que se deben efectuar en el transcurso del proyecto.

  - **Estructura de la memoria:** Se desarrolla una pequeña explicación de cada uno de los puntos que forman este trabajo.

- **Capítulo 2. Estado del arte:** En este capítulo, se explica el estado actual del mercado para la implementación de una red 4G y 5G. Además de incluir las tecnologías de IoT como NB-IoT y LTE-M, consta de:

  - **Alternativas tecnológicas para la implementación de 4G y 5G:** Se listan las diferentes opciones para implementar 4G y 5G para un entorno de investigación.
  - **Conclusiones sobre el software elegido:** Se comparan todas las herramientas y se indican los motivos de la alternativa que ha sido elegido.

- **Capítulo 3. Fundamentos teóricos:** Se describe toda la parte teórica en la que se basa el proyecto y tecnologías usadas. Se divide en:

  - **Tecnología móvil 4G:** Se detallan los principales conceptos teóricos, haciendo hincapié en su arquitectura, pila de protocolos y espectro de 4G.
  - **Tecnología móvil 5G:** Al igual que en 4G se describe los principales conceptos teóricos, haciendo hincapié en su arquitectura, pila de protocolos y espectro, pero en este caso de la quinta generación (5G).
  - **Tecnología LTE-M:** De forma similar a 4G y 5G se explican sus características principales junto a su pila de protocolo y arquitectura. Además de las frecuencias y espectro utilizadas.
  - **Tecnología NB-IoT:** Se describe de igual forma que LTE-M (*Long Term Evolution categoría M1)*, incluyendo los mismos puntos.

- **Capítulo 4. Planificación y coste:** Se indica el coste y los recursos utilizados en el transcurso del proyecto. Se divide en:

  - **Planificación temporal:** Se describe el orden y el tiempo utilizado en la realización de las tareas.
  - **Recursos:** Se exponen los recursos humanos, hardware y software usados en el proyecto.
  - **Presupuesto total:** Mediante el uso de tablas se detalla un presupuesto con los gastos realizados.

- **Capítulo 5. Configuración y pruebas:** En esta sección, se explica la instalación de Amarisoft junto a la configuración realizada en cada uno de los archivos de los componentes para cada tecnología móvil, la configuración de los dispositivos utilizados y las pruebas efectuadas.

  - **Instalación y configuración del software de Amarisoft:** Se desarrollan los pasos que se deben de seguir para la instalación del software Amarisoft. Además se explica la configuración de los archivos de los componentes y los dispositivos.
  - **Pruebas realizadas:** Se exponen las pruebas realizadas en el proyecto, diferenciándolas entre 4G/5G y LTE-M/NB-IoT.

- **Capítulo 6. Análisis de resultados:** Se analizan y describen los resultados obtenidos de las pruebas realizadas para las cinco tecnologías.
  - **Análisis 4G y 5G:** Se analizan los resultados obtenidos a partir de iPerf y ping. Comparando para cada tecnología de forma individual las diferencias que se obtiene en rendimiento y latencia dependiendo de los parámetros seleccionados.
  - **Análisis entre tecnologías 4G y 5G:** En esta sección se comparan las tecnologías de 4G y 5G de forma conjunto, obteniendo conclusiones sobre que tecnología tiene un mejor rendimiento o una menor latencia.
  - **Análisis LTE-M y NB-IoT:** Se analizan los menajes enviados al realizar la prueba de conectividad, así como alguna comparativa entre ambas tecnologías del *throughput* y latencia.

- **Capítulo 7. Conclusiones y líneas futuras:** En esta última sección, se exponen las conclusiones obtenidas tras la realización del trabajo y las líneas a futuro que se pueden seguir para mejorar este proyecto o para la realización de uno nuevo pero siguiendo el hilo de este.
  - **Conclusiones**: Se presentan las conclusiones adquiridas tras terminar el proyecto.
  - **Líneas futuras**: Se definen los nuevos trabajos a futuro que se pueden realizar siguiendo la línea de este proyecto.

Capítulo 2

Estado del Arte

En este capítulo, se llevará a cabo el estado del arte donde se exponen las diferentes alternativas que existen en el mercado para diseñar, implementar y analizar tanto una red 4G como 5G. A su vez identificar que opción permite desplegar redes para IoT, en concreto las tecnologías de LTE-M y NB-IoT.

2.1. Alternativas Tecnológicas para la implementación de 4G y 5G

En este apartado, se exploran las diversas maneras de implementación de las tecnologías móviles 4G/5G, siendo este desde un entorno de experimentación.

Los distintos *software* se clasifican según su implementación. Existen algunas opciones que implementan únicamente la red troncal o la parte radio. Otras alternativas pueden desplegar una implementación completa es decir, tanto la parte radio como la parte troncal.

2.1.1. Free5GC

Free5GC [7] se trata de un proyecto de código abierto con una licencia totalmente gratuita, para tecnologías móviles 5G, implementa la parte troncal de la red (5GC), tal como se define en el *Release 15* [8] de 3GPP, pero no solo se limita a este sino que l a Universidad Nacional Chiao Tung (NCTU), desarrolló este proyecto y busca adaptarse a las progresivas actualizaciones que realiza 3GPP.

Actualmente, se compone de 3 etapas, en su versión más reciente cuenta con un despliegue completo de 5GC, con funciones cómo N3IWF(*Non-3GPP Inter Working Function*) para habilitar la comunicación con redes no-3GPP

como Wi-Fi y redes por cable o ULCL (*UP Uplink Classifier*) para clasificar y gestionar el tráfico que va desde el móvil hacia la red (*uplink*). En la Figura 2.1 se muestra la arquitectura utilizada en la segunda etapa. Destacar que al ser un proyecto centrado en 5G no es capaz de implementar tecnologías como NB-IoT.

El código fuente de su última versión se puede descargar desde su página oficial de *github* [9], donde se indican los pasos para su instalación y configuración. [10].

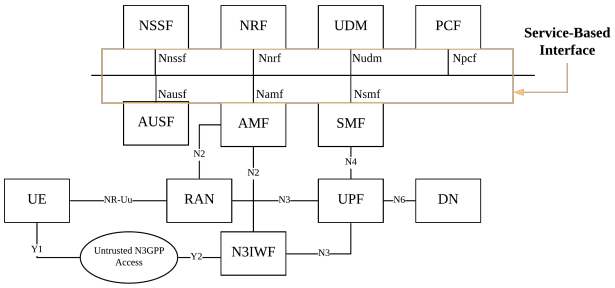

Figura 2.1: Arquitectura Free5GC en la etapa 2 [7]

2.1.2. Open5GS

Open5GS [11] al igual que Free5GC, se trata de un proyecto de código abierto, pero en este caso, usa el lenguaje C. Implementa LTE (EPC) además del núcleo de la red 5G (5GC) según se describe en el *Release 17* [12] de 3GPP. En su página de GitHub [13] se pueden encontrar soluciones a bastantes problemas típicos, así como los pasos a seguir para su instalación y configuración.

Al adentrarse en la arquitectura básica del software [14] se puede comprobar en la Figura 2.2 que cuenta con un núcleo 4G/5G NSA (*Non-Stand Alone*) compuesto por los componentes descritos en la Tabla 2.1, además de funciones de red para implementar un *core* 5G SA (*Stand Alone*). El núcleo se divide en el plano de control y el plano de usuario que se encuentran separados físicamente. En cuanto al núcleo de 5G SA funciona de manera diferente al de 4G, se basa en una arquitectura basada en servicios (SBA), provocando que el plano de usuario sea más simple. Estas funciones las podemos ver en la parte de 5G en la Tabla 2.1.

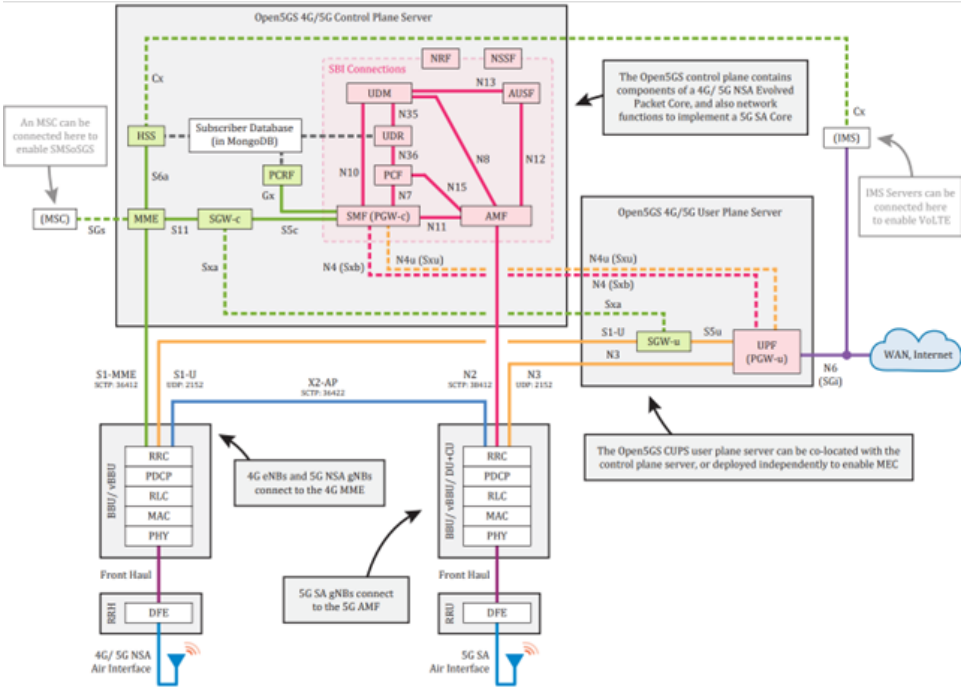

Figura 2.2: Arquitectura Open5GS [14]

2.1.3. UERANSIM

Al igual que los dos proyectos anteriores, UERANSIM también es un software de código abierto pero al contrario de los demás, es un simulador de 5G UE y RAN basándose en 3GPP [15].

En cuanto a las funcionalidades que presta, destaca que la interfaz radio es simulada sobre el protocolo UDP (*User Datagram Protocol*). Se divide en el plano de control y el plano de usuario, este primero se separa en NAS (*Non-Access-Stratum Protocol*) que tiene el control de UE y NGAP a su vez de RAN, el plano de control es implementado por el protocolo GTP(*GPRS Tunnelling Protocol*) y solo admite IPv4. Por tanto, esta herramienta únicamente se centra en la simulación de la parte radio de 5G.

2.1.4. OpenAirInterface

OpenAirInterface [16] se trata de un software abierto, al igual que los proyectos anteriores, pero en cambio, puede implementar la red de acceso por radio (RAN) y la red central (CN). Consiguiendo desplegar una red completa de 5G. También es compatible con la tecnología de LTE, LTE-M y NB-IoT, siendo una opción verdaderamente prometedora para la realización

4G Componentes	Definición
MME	Entidad de Gestión de la Movilidad
HSS	Servidor de suscriptores doméstico
PCRF	Función de políticas y reglas de cobro
SGWC	Plano de control de puerta de enlace de servicio
SGWU	Plano de usuario de puerta de enlace de servicio
PGWC/SMF	Packet Gateway Control Plane
PGWU/UPF	Packet Gateway User Plane

5G Funciones	Definición
NRF	Función de repositorio NF
SCP	Proxy de comunicación de servicio
AMF	Función de gestión de acceso y movilidad
SMF	Función de gestión de sesiones
UPF	Función de plano de usuario
AUSF	Función del servidor de autenticación
UDM	Gestión unificada de datos
UDR	Repositorio Unificado de Datos
PCF	Política y función de carga
NSSF	Función de selección de segmento de red
BSF	Función de soporte de enlace

Tabla 2.1: Componentes de 4G y Funciones de 5G

de este proyecto. Es desarrollado por la OSA (*OpenAirInterface Software Alliance*) y su misión es facilitar el uso de este software.

Tienen diferentes proyectos [17], pero nos centraremos solo en dos, que nos ayudan a desplegar la parte de red de acceso radio (RAN) y la red central (CN).

En el proyecto OAI 5G RAN, el objetivo es crear un software 5G que admita gNB Non-Stand-Alone y Stand-Alone , 5G NSA y SA UE como aparece en la Figura 2.3.

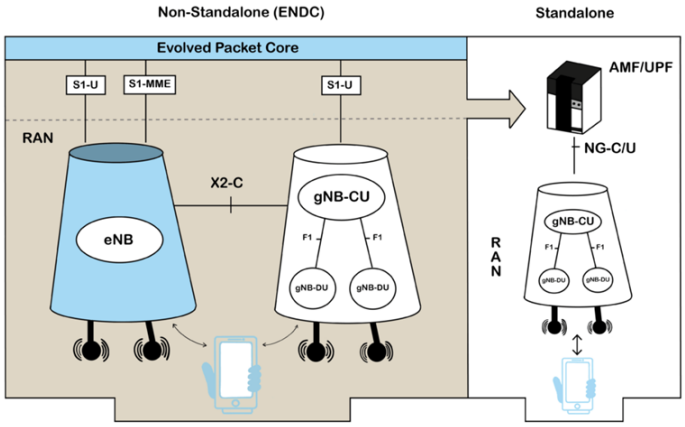

Figura 2.3: Arquitectura RAN OpenAirInterface [17]

En el proyecto de OAi-5G Core Network (CN), se implementa 5G (SA) con un amplio abanico de funciones para que sea compatible con 3GPP. Al igual que Open5GS, se basa en una arquitectura basada en servicios, separando el plano de control del plano de usuario, como se muestra en la Figura 2.4. En la última versión v1.4.0, se implementa algunas de las funciones como las descritas anteriormente para Open5GS en concreto, AMF, SMF, UPF, NRF, AUSF, UDM, UDR y NSSF.

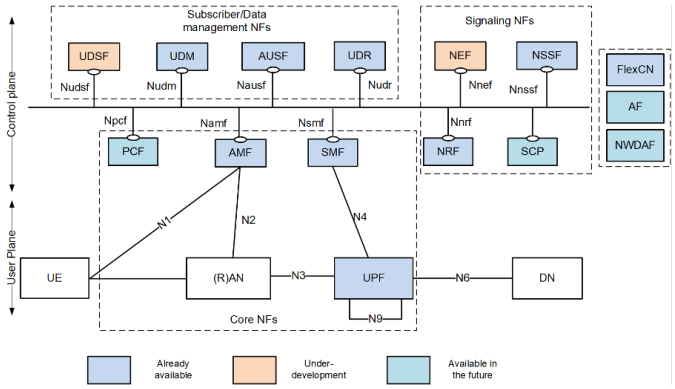

Figura 2.4: Arquitectura CN OpenAirInterface [17]

2.1.5. SrsRAN

SrsRAN [18] es un proyecto que cuenta con un software comercial de código abierto para implementar 4G, 5G. Se desarrolló completamente por *Software Radio Systems* (SRS) mediante el lenguaje C++ y se ha planificado de forma que sea modular, portable, modificable y escalable para poder usarse desde una raspberry a un centro de datos, tanto para 5G como para su predecesor (4G) [19].

Actualmente, cuentan con dos proyectos, srsRAN para 4G [20] en la versión 22.10 y srsRAN Project, el principal en el que están trabajando, para 5G [21].

Con el proyecto para 4G, se puede crear una red 4G extremo a extremo, como se observa en la Figura2.5, los elementos que incluye son:

- **srsUE:** Implementa una aplicación UE 4G con características de 5G.
- **srsENB:** Se trata de la implementación de una estación base Evolved Node B (eNB), para conectar al usuario (UE) con la EPC.
- **srsEPC:** Implementación del núcleo de la red LTE, que cuenta con MME, HSS, S-GW y PGW.

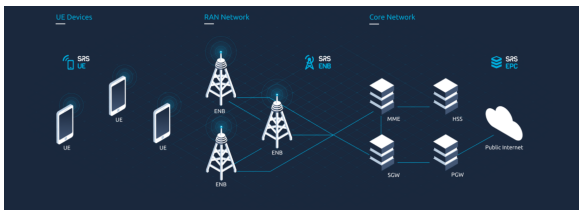

Figura 2.5: Arquitectura 4G SrsRAN [20]

Con el proyecto para 5G [22], se desarrolla 5G CU/DU, solución para capa 1, 2 y 3, donde cumple con las especificaciones de 3GPP. Separando las funcionalidades entre la CU (usuario centralizado) y la DU (usuario distribuido), como se muestra en la Figura 2.6.

2.1.6. Amarisoft

Amarisoft [23] es una empresa francesa fundada en 2012, que desarrolla tecnología y soluciones a un precio accesible pero de alta calidad, tanto a nivel comercial como en el ámbito del desarrollo e investigación del 4G/5G [24].

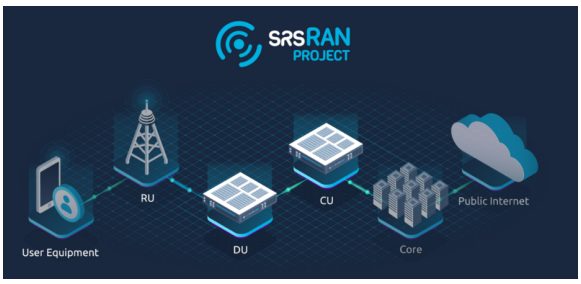

Figura 2.6: Arquitectura 5G SrsRAN [21]

Su software Amarisoft, de extremo a extremo, cumple con las especificaciones 3GPP, sus principales elementos son los mostrados en la Figura 2.7, donde cabe destacar eNB, EPC, gNB, 5GC y simulador UE [25]:

- **eNB:** Se trata de la estación base de LTE, compatible con LTE-Advanced y MIMO. También puede desplegar tecnología IoT, tanto NB-IoT y LTE-M. Soporta todas las bandas de frecuencia TDD y FDD.
- **gNB:** Al igual que eNB hace alusión a la estación base pero en este caso para NR. Soporta el rango de frecuencia de Fr1 y FR2, además del tipo de duplexación (TDD o FDD). Permite MIMO y los modos de 5GNSA y5G SA.
- **EPC:** *Core* para la tecnología de 4G y 5G NSA. Amarisoft consigue compactar en esta parte troncal los elementos de MME, S-GW, PGW y HSS.
- **5GC:** Parte troncal para la tecnología de 5G. Llegando a soportar tanto NSA como SA. Consigue compactar las funciones AMF, AUSF, SMF y UE.
- **Simulador UE:** Simulador de usuarios que permite crear varios UEs a la vez, compartiendo un mismo espectro. Soportando MIMO, diferentes duplexaciones y rangos de frecuencias. Es capaz de simular usuarios para las tecnologías de 5G SA, 5G NSA, LTE,LTE-Advanced, NB-IoT y LTE-M.

Nuestro estudio se centrará en los productos de investigación de Amarisoft, a través de Amari Callbox Series [26]. En nuestro caso se empleará un ordenador de propósito general junto al software de Amarisoft, consiguiendo una funcionalidad similar. Puede actuar como eNB/gNB y EPC/5GC siendo capaz de implementar 5G NSA y SA, LTE, LTE-M y NB-IoT. En definitiva, se puede desplegar una red completa de 4G, 5G y compatible con IoT.

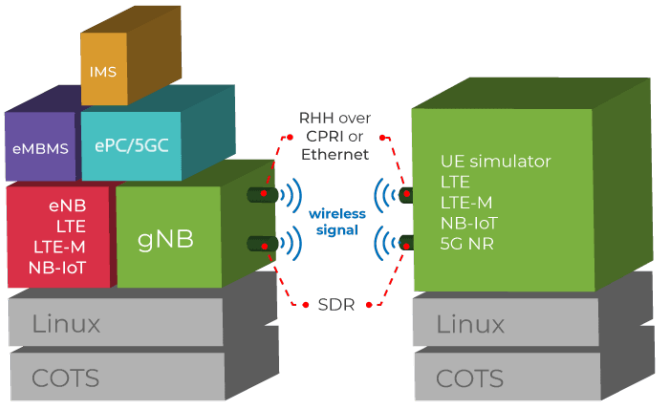

Figura 2.7: Elementos Amarisoft [25]

2.1.7. Conclusiones sobre el software elegido

En este apartado, se muestra la Tabla 2.2, se trata de una comparativa de las opciones descritas en el punto anterior. Se dividen dependiendo del tipo de tecnología móvil que son capaces de desplegar, así como si pueden implementar una red completa.

Software	**Tecnologías que puede desplegar**				**Implementación completa**	
	LTE (4G)	5G-NSA	5G-SA	LTE-M Nb-Iot	Parte radio	Parte troncal
Free5GC	X	X	X			X
Open5GS	X	X	X	X		X
UERANSIM	X	X	X		X	
OpenAirInterface	X	X	X	X	X	X
SrsRAN	X	X	X		X	X
Amarisoft	X	X	X	X	X	X

Tabla 2.2: Comparativa de alternativas para implementación de red

Destacar que todas las opciones son de código abierto excepto la última (Amarisoft), que tiene un coste adicional al tratarse de un software privado.

Las opciones Free5GC, Open5GS y UERANSIM mostradas en la Tabla 2.2 no implementan la red completa (parte radio y parte troncal) y uno de los objetivos de este trabajo es el despliegue total para su posterior análisis. El software SrsRAN, implementa la red troncal y radio tanto LTE como 5G, pero no soporta tecnologías IoT como LTE-M y NB-IoT. Por tanto, estas herramientas están descartadas para su utilización en el proyecto.

Únicamente se cuenta con dos alternativas que sean capaces de realizar implementaciones completas de las tecnologías móviles 4G, 5G y de las tecnologías IoT de forma individual, son OpenAirInterface y Amarisoft. Pero, al combiar el software de srsRAN junto a Open5GS también se podría obtener el objetivo buscado. Por tanto, se cuenta con tres opciones que cumple con el objetivo, OpenAirInterface, Amarisoft y srsRAN junto a Open5GS. Tras analizarlas, se comprende que la opción que cuenta con mayores funcionalidades y facilidad de medición se trata de Amarisoft. Al ser una versión comercial es mucho más estable y mejor optimizada que las demás opciones. Uno de los inconvenientes es el elevado precio de su licencia, además del costo de los recursos *hardware* que se deben utilizar. Pero incluye una atención al cliente rápida y un soporte para solucionar cualquier problema que se tenga con su software, algo de gran ayuda en el desarrollo del proyecto.

Capítulo 3

Fundamentos Teóricos

En este capítulo, se desea dar una visión general sobre los conocimientos teóricos necesarios para la realización del proyecto y el despliegue de las diferentes tecnologías móviles. Empezando por una explicación de 4G y 5G, donde contará con aspectos como las características y componentes principales, pila de protocolos, espectro y frecuencia. Por último, se expondrá de igual forma las tecnologías LTE-M y NB-IoT.

3.1. Tecnología móvil 4G

La tecnología móvil 4G se desplegó con el estándar para comunicaciones inalámbricas LTE de 3GPP, se introdujo en el *Release 8* [27] y sus mejoras terminan en el *Release 13* [5]. Se desarrolló para satisfacer la creciente demanda de ancho de banda necesitada por los usuarios [28].

Esta tecnología se basa en IP (*Internet Protocol*), lo que creó una base para 5G. Una de sus mayores diferencias es la simplicidad de su arquitectura, provocando una red menos costosa y una mayor velocidad de transmisión. Teóricamente, en su inicio, la tasa de transmisión es de 100 Mbps para móviles y 1 Gbps para usuarios estacionarios, pero posteriormente en el *Release 10* [29] surgió LTE-Advanced proporcionando una mayor velocidad y un mejor rendimiento al introducir la tecnología *Multiple-Input Multiple-Output* (MIMO), *Self-Organizing Networks* (SON), la agregación de portadora y los *relay* [30] [31] como se puede ver en la Figura 3.1. Entrando en más detalles las características principales de 4G son [32]:

- **Velocidad:** En el primer *release*, se admite una velocidad de 300 Mbps para *downlink* y 75 Mbps para *uplink*.
- **Latencia:** Una latencia por debajo de 10 ms y un tiempo para establecer la conexión de 300 ms.

- **Capacidad y cobertura:** Cuenta con una mayor área de cobertura, además de poder soportar un mayor número de dispositivos.

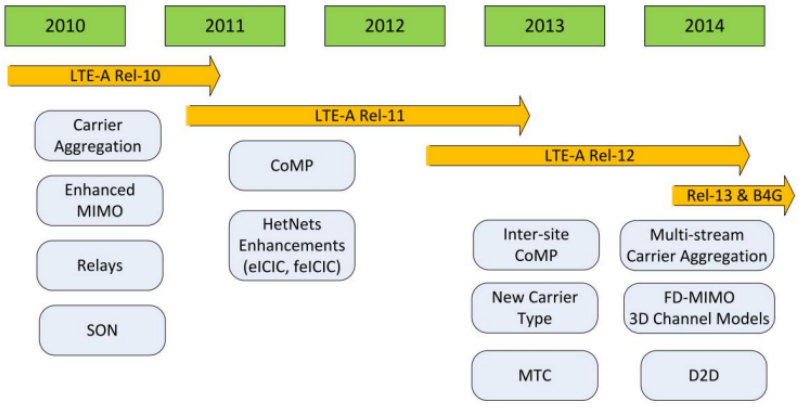

Figura 3.1: Roadmap LTE-*Advanced* [30]

3.1.1. Arquitectura de red 4G

La arquitectura de 4G se divide principalmente en dos partes, en la red de acceso (E-UTRAN), sirve para conectar al usuario con la red troncal (EPC), la responsable de transferir información.

- **Red de Acceso, Enhanced UMTS Terrestrial Radio Access Network (E-UTRAN) [33]:** Se trata de la red de acceso radio (RAN). Gracias a ella se consigue la comunicación entre la EPC y los UE. Está formada por las estaciones base que pasan a llamarse *evolved NodeB* (eNB), usando tecnología MIMO y *Orthogonal Frequency-Division Multiplexing* (OFDM). Además, la red de acceso se simplifica ya que las estaciones base incorporan funcionalidades que pertenecían a las *Radio Network Controller* (RNC) o *Base Station Controller* (BSC) de 3G, haciendo que no se necesite este tipo de nodos de control y permitiendo con ello la comunicación IP entre los diferentes eNB.

- **Red Troncal, Evolved Packet Core (EPC):** En esta parte de la red una de las grandes diferencias con 3G es la integración de voz y datos en una única red de datos mediante IP. La comunicación con la red de acceso se realiza mediante dos interfaces, el plano de control para señalización y el plano de usuario para datos, como se observa en la Figura 3.2.

Los componentes más importantes son:

- **Mobility Management Entity (MME):** Realiza el encaminamiento y transferencia de las comunicaciones asignándole a cada UE una pasarela S-GW (*Service Gateway*). Además, gestiona el plano de control,localización y la autenticación de los dispositivos.
- **Service Gateway (S-GW):** Esta entidad se trata de la pasarela para el plano de usuario para trasladar el tráfico entre redes. La asigna la MME, dependiendo de la carga de la red y la localización geográfica.
- **Home Suscriber Server (HSS):** Pertenece al plano de control, se trata de una base de datos que almacena la información de los clientes de un operador, como localización, datos técnicos y los permisos de acceso a los servicios.
- **Packet Data Network Gateway (PGW):** La comunicación del plano de usuario que vienen del S-GW pasan por esta entidad. Realiza la interconexión entre diferentes redes, destacando que puede filtrar o monitorizar el tráfico para encargarse de la tarificación. Asimismo, la red cuenta con una única S-GW, pero puede contar con más de una PGW, ya que puede estar interconectada con varias redes.

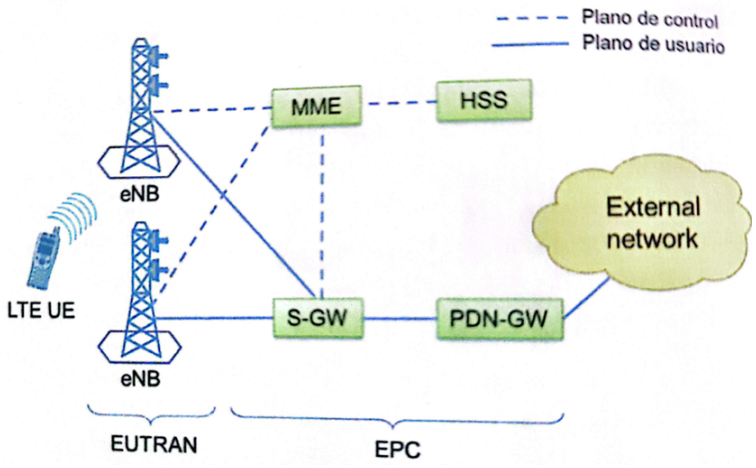

Figura 3.2: Arquitectura red 4G [33]

3.1.2. Pila de Protocolos de 4G

Una forma sencilla a la hora de hablar de los protocolos que utiliza la tecnología móvil 4G es dividirlos según la función que realizan en las capas del modelo OSI. En nuestro caso, nos centraremos en las primeras capas,

la capa Física (Capa 1), en la de enlace de Datos (Capa 2) y en la de Red (Capa 3) [34] [35].

- **Capa física:** Se utilizan diferentes protocolos dependiendo del tipo de enlace empleado (enlace descendiente o ascendente). Para el enlace descendente se usa *Orthogonal Frequency Division Multiple Access* (OFDMA), se trata de una técnica de multiplexación que permite en un mismo ancho de banda la transmisión simultanea de varios usuarios. Se debe a que al contrario de las modulaciones multiportadoras clásicas como *Frequency Division Multiplexing* (FDM) y *Time Division Multiplexing* (TDM), divide el espectro en múltiples subportadoras que se superponen en el tiempo, permitiendo que varios usuarios puedan compartir una misma subportadora haciendo que OFDMA sea muy eficiente en la utilización del espectro, como se muestra en la Figura 3.3. Para el enlace ascendente se emplea *Single Carrier Frequency Division Multiple Access* SC-FDMA, es similar a OFDMA, pero utilizando modulación de portadora única.

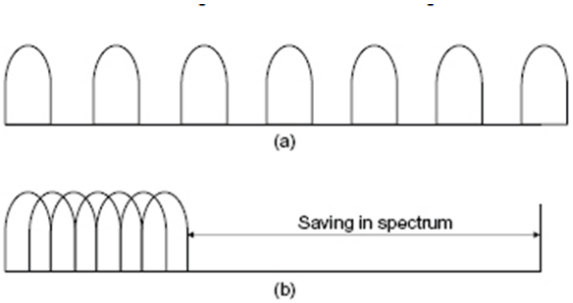

Figura 3.3: Comparativa OFDMA con modulaciones multiportadoras clásicas [34]

- **Capa de enlace de datos:** Como se ve en las Figuras 3.4 3.5, esta capa se divide en 3 tanto para el plano de control como para el plano de usuario. *Media Access Control* (MAC), este protocolo permite la conexión a RAN. *Radio Link Control* (RLC), su principal función es la segmentación y su posterior reensamblaje de los paquetes de datos. *Packet Data Convergence Protocol* (PDCP), realiza comprensión, descompresión de cabeceras IP y transferencia de datos.

- **Capa de red [36]:** Para el plano de control se utiliza *Radio Resource Control* (RRC), permitiendo conectar eNB con UE. Además, destaca el protocolo IP, para enrutamiento.

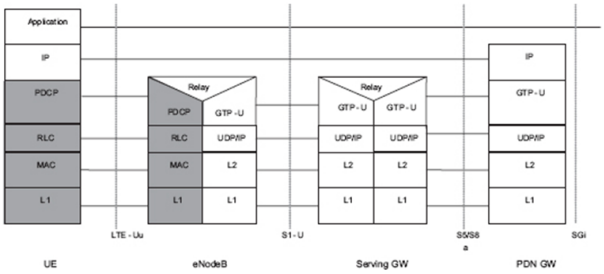

Figura 3.4: Pila de protocolos del plano de usuario [34]

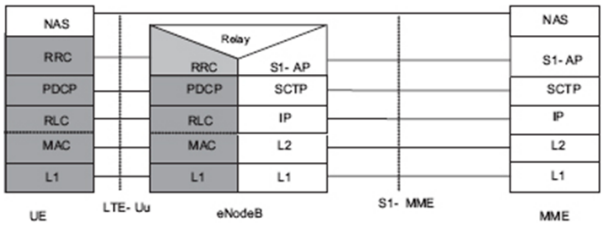

Figura 3.5: Pila de protocolos del plano de control [34]

3.1.3. Espectro y Frecuencia 4G

El estándar de LTE, permite la utilización de diferentes técnicas de duplexación, FDD para frecuencias, permitiendo recibir y emitir datos a la vez, usando distintas bandas de frecuencia. TDD para tiempo, pudiendo transmitir utilizando una sola banda de frecuencia, separada por intervalos de tiempo [37].

En la tabla 3.1 se muestran las bandas de frecuencias utilizadas por 4G en el Modo FDD (*Frequency Division Duplex*). En estas tablas, solo se recogen las bandas de frecuencias que no están obsoletas y que se utilizan en España, es decir, que su región sea Global, EMEA (*Europe the Middle East and Africa*) o EU [38].

Además, en la tabla 3.2 se indican las frecuencias para el modo TDD, donde la banda 34 y 38 pertenece a EMEA, mientras que las demás son globales.

LTE Band	Modo	Uplink (MHz)	Downlink (MHz) (EARFCN)	Bandwidth DL/ UL (MHz)
1	FDD	1920-1980	2110-2170 (300)	60
3	FDD	1710-1785	1805-1880 (1575)	75
7	FDD	2500-2570	2620-2690 (3100)	70
8	FDD	880-915	925-960 (3625)	35
20	FDD	832-862	791-821 (6300)	30
22	FDD	3410-3490	3510-3590 (7000)	80
28	FDD	703-748	758-803 (9435)	45
31	FDD	452.5-457.	462.5-467.55 (9895)	5
65	FDD	1920-2010	2010-2200 (65986)	190
72	FDD	451-45	461-4666 (68961)	5

Tabla 3.1: Bandas de frecuencia de 4G modo FDD [38]

LTE Band	Modo	Downlink (MHz) (EARFCN)	Bandwidth DL/ UL (MHz)
34	TDD	2010-2025 (36275)	15
38	TDD	2570 - 2620 (38000)	50
41	TDD	2496 - 2690 (40620)	194
46	TDD	5150-5925 (50665)	775
47	TDD	5855-5925 (54890)	70
48	TDD	3550-3700 (55990)	150

Tabla 3.2: Bandas de frecuencia de 4G modo TDD [38]

3.2. Tecnología móvil 5G

La tecnología móvil 5G fue definida en el *Release 15* [8] de 3GPP y se completó en el 2019. Tiene una velocidad y capacidad mayor que su predecesor y se empezó a desarrollar para ser utilizada en varios sectores de la industria, aplicaciones críticas, y aplicaciones IoT, que necesitan un gran número de dispositivos conectados y una baja latencia. Las características de 5G son [39]:

- **Velocidad:** Obteniendo una tasa de transferencia 10 veces mayor que 4G, llegando incluso a 10Gbps.
- **Latencia:** Una de las principales características de 5G, llegando a 1 ms y así poder realizar actividades a tiempo real como telemedicina.
- **Capacidad y cobertura:** Llegando a una capacidad y cobertura mucho mayor que 4G.

Los principales servicios que mejora 5G a 4G son [40][41]:

- **Banda ancha móvil mejorada (eMBB):** Especificados para velocidades altas. Ofreciendo hasta 10 Gbps permitiendo descarga rápida. Puede dar cobertura a una gran cantidad de situaciones, tanto exteriores como interiores y de alta movilidad.
- **Comunicaciones ultra confiables y baja latencia (URLLC):** Para aquellas situaciones que necesiten una alta disponibilidad llegando a 99.999 % y una latencia lo más baja posible, inferiores a 1 ms. Para aplicaciones críticas como la telemedicina o transporte autónomo.
- **Comunicaciones masivas tipo máquina (mMTC):** Para la industria del IoT, se necesitan que los sistemas 5G admitan grandes densidades de dispositivos y una gran escalabilidad. Llegando a soportar hasta 1 millón de dispositivos por Km^2 simultáneamente.

En la figura 3.6 se puede observar un esquema con los servicios de 5G.

3.2.1. Arquitectura de red 5G

La arquitectura de la tecnología 5G se divide en dos, al igual que 4G. La red acceso en este caso se llama 5G RAN y la red troncal 5G, 5GC. Para adentrarse en los diferentes componentes que forman cada parte de esta tecnología, se debe destacar que una red 5G se puede implementar de dos formas diferentes [40]:

- **Arquitectura Non-Stand Alone (NSA):** En este caso, la red de acceso 5G (5G RAN) y su interfaz radio, *New Radio* (NR) se usa junto

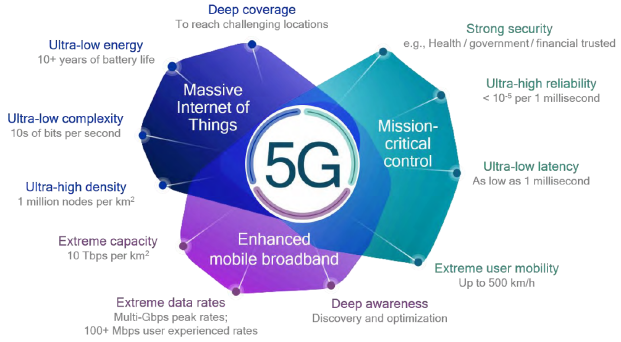

Figura 3.6: Servicios 5G [42]

con la red troncal de la tecnología móvil 4G (EPC), como se observar en la Figura 3.7. Provocando que la tecnología NR se encuentre disponible sin tener que relevar la infraestructura ya disponible. Con este tipo de arquitectura solo se tienen los servicios de 4G junto con las capacidades de 5G NR. La conexión de plano de control con el EPC se realiza a través de uno de los nodo gNB o eNB mientras el plano de usuario es compartido por ambos.

- **Arquitectura Stand-Alone (SA):** Se trata de la arquitectura completa de 5G, donde la 5G-RAN, compuesta por gNB se conectan a la red troncal de 5G (5GC), en la Figura 3.8 se muestran dichos componentes. Los tres principales servicios de 5G se implementan perfectamente con este caso.

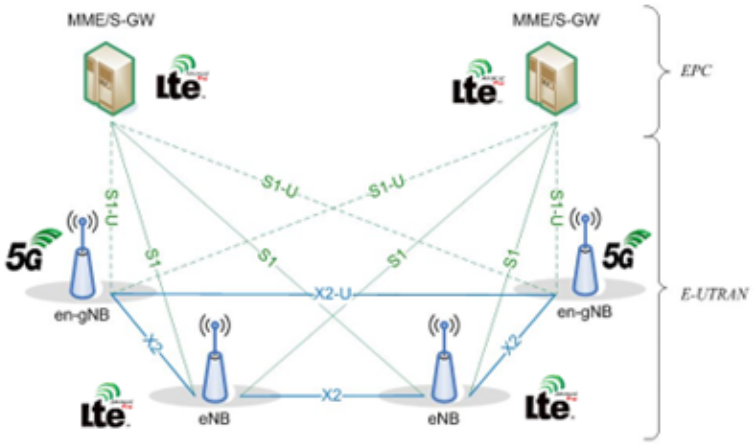

Figura 3.7: Arquitectura Non-Stand Alone (NSA) [40]

Como se ha indicado, las dos principales partes de la tecnología 5G son:

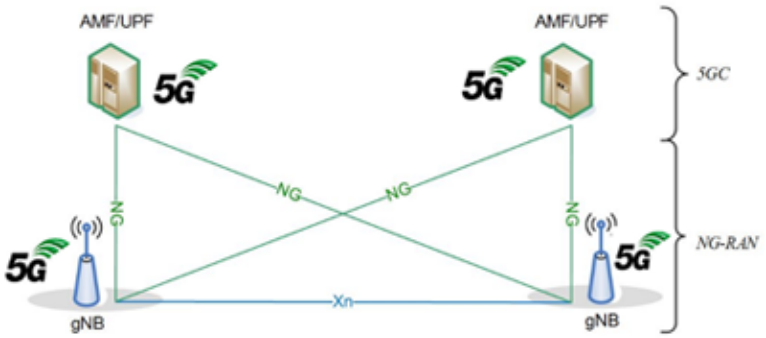

Figura 3.8: Arquitectura Stand-Alone (SA) [40]

- **Red de Acceso Radio 5G (5G-RAN) [40][43]:** Su principal función es similar a la de red de Acceso de 4G, comunicar al equipo del usuario con la red troncal. El componente más importante es el nodo gNB, da servicio para el plano de usuario y plano de control.
- **Red Troncal (5GC) [40][43]:** Es la parte central de 5G, donde conecta la red de acceso con otra red. Se basa en la Arquitectura basada en servicios (SBA), que da un enfoque de modularidad y reutilización. Ya que cada elemento que compone esta red se define como Funciones de Red (NF) y puede dar su servicio a cualquier de las otras NF. En la Figura 3.9, se muestran las principales funciones de red que componen 5GC, en la parte superior las funciones de red del plano de control mientras que en la inferior los NF usados para el transporte de los datos de usuarios, donde solo pertenece a la red troncal el UPF.

    - **User Plane Function (UPF):** Es un punto clave, ya que se conecta con la red de datos (DN) mediante la interfaz N6, haciendo de punto de interconexión. Se encarga de los datos de usuario, procesándolos y reenviándolos, mientras es controlado por el SMF.
    - **Access and Mobility management Function (AMF):** Accede al UE y a la RAN. Su función es gestionar el registro, la conexión y la movilidad de los UE. Se conecta con estos con la interfaz N1 y con la red de acceso con la N2.
    - **Session Management Function (SMF):** Gestiona las sesiones de usuarios, estableciendo, modificando y liberando las sesiones. Asignándoles una IP a cada sesión. Se une con la UPF por la interfaz N4.
    - **The Network Slice Selection Function (NSSF):** Selecciona la instancia de Network Slice que sirve al UE. Permitiendo que cada UE pueda usar varias.

- **Network Exposure Function (NEF):** Administra datos de la red y ayuda a aplicaciones a acceder a datos de 5GC.
- **Network Repository Function (NRF):** Da soporte al servicio de registro para controlar las demás NF.
- **Policy Control Function (PCF):** Controla el tráfico de los usuarios no supere las capacidades.
- **Unified Data Management (UDM):** Base de datos de los suscriptores del operador, función similar al HSS de 4G.
- **Authentication Server Function (AUSF):** Proporciona la autenticación de los diferentes dispositivos.
- **Application Function (AF):** Su función es la de un servidor, para interaccionar con NF del plano de control.

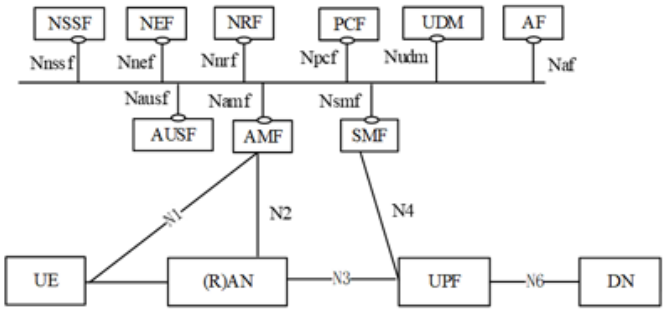

Figura 3.9: Arquitectura 5GC [40]

3.2.2. Pila de Protocolos de 5G

Al igual que la pila de protocolos de 4G, en 5G se organiza en las mismas capas. A continuación, se indicarán las tres primeras capas junto a los protocolos correspondientes [39][41].

- **Capa física:**: Tanto para el enlace ascendente como descendente utiliza OFDM básico.
- **Capa Enlace de datos:** : Utiliza los mismos tipos de protocolos que 4G, MAC, RLC y PDCP, añadiendo nuevas funciones.
- **Capa red:** : Al igual que 4G, se sigue utilizando el protocolo IP.

En conclusión, la pila de protocolos de 5G es similar a la de su tecnología móvil predecesora.

3.2.3. Espectro y Frecuencia 5G

Con el escenario de 5G que necesita una alta capacidad y velocidad, se implementaron dos bandas de frecuencias en 5G, sub-6 GHz (FR1) y las milimeter-wave (FR2) por encima de 24GHz. Exactamente el rango de frecuencias de FR1 es 410-7125 MHz, mientras que el de FR2 es 24,25-52,6 GHZ, pudiendo aumentar o complementarse en nuevas versiones de 3GPP [39].

Como en 4G, se mostrarán en las tablas 3.3 y 3.4, las bandas de frecuencia utilizadas en España, en las regiones Global, EMEA o EU. En primer lugar, para FR1 y posteriormente para el rango de frecuencias FR2 [44].

NR Band	**Mode**	**Uplink (UL) (MHz)**	**Downlink (MHz) nr-arfcn**	**Bandwidth DL/ UL (MHz)**
n1	FDD	1920 - 1980	2110 - 2170 (428000)	60
n3	FDD	1710 - 1785	1805 - 1880 (368500)	75
n7	FDD	2500 - 2570	2620 - 2690 (531000)	70
n8	FDD	880 - 915	925 - 960 (188500)	35
n20	FDD	832 - 862	791 - 821 (161200)	30
n28	FDD	703 - 748	758 - 803 (156100)	45
n34	TDD	-	2010-2025 (403500)	15
n38	TDD	-	2570 - 2620 (519000)	50
n41	TDD	-	2496 - 2690 (518598)	194
n78	TDD	-	3300 - 3800 (636666)	500

Tabla 3.3: Bandas de frecuencia de 5G para FR1 [44]

3.3. Tecnología LTE-M

Se desarrolló LTE-M [45] para dar soporte al internet de las cosas, en comunicaciones de tipo máquina ya que LTE no soporta las comunicaciones mMTC. Sus características principales eran admitir dispositivos de bajo costo, gran cantidad de dispositivos, mejor cobertura en 15 dB, con una

NR Band	Modo	Uplink (UL) (MHz)	Downlink (MHz) nr-arfcn	Bandwidth DL/ UL (GHz)
n257	TDD	28 GHz	26500 - 29500 (2079166)	3
n258	TDD	26 GHz	24250 - 27500 (2043750)	3.25
n259	TDD	41 GHz	39500 - 43500 (2304166)	4
n260	TDD	39 GHz	37000 - 40000 (38500)	3

Tabla 3.4: Bandas de frecuencia de 5G para FR2 [44]

duración de batería de 10 años y una latencia entre 15ms y 30ms.

Para poder conseguir estas propiedades, se introdujo un nuevo tipo de UE, con un ancho de banda de 1.4 MHz, una sola antena receptora, un tamaño máximo de bloque de transporte de 1000 bits y funciona en semidúplex. Además, se incluyen funciones adicionales como eDRX (*Extended Discontinuous Reception* para que los dispositivos se encuentren en estado inactivo una gran parte del tiempo mientras esperan datos, provocando una bajada de consumo de energía.

3.3.1. Arquitectura y pila de protocolos LTE-M

Al basarse en LTE su arquitectura es similar a la descrita en la Figura 3.2 y únicamente se necesita actualizar las estaciones bases de 4G para admitir LTE-M. En cuanto a su la pila de protocolos, se pueden observar algunas diferencias, sobre todo en la capa física y de enlace, aunque también en algunas capas superiores [46].

- **Capa física:** LTE-M emplea una modulación diferente a LTE, tanto para el enlace descendente como ascendente utiliza QPSK (*Quadrature Phase Shift Keying*), es una modulación más sencilla y robusta en medios con poca señal o muchas interferencias. ADemás utiliza un canal GSM(*Global System For Mobile Comunication*) de 200 KHz.

- **Capa de enlace:** Esta tecnología al estar pensada para IoT utiliza tramas de datos más pequeñas para el ahorro de energía y optimización de espectro. Además, los protocolos de control para señalización o encabezados en LTE-M utilizan menos recursos para su gestión en comparación con LTE.

3.3.2. Espectro y frecuencia LTE-M

En cuanto al espectro, esta tecnología puede utilizar cualquier banda de LTE definidas en 3.1.3, para ello, se definen canales especiales y procedimientos en las celdas para limitar el ancho de banda dando con ello una utilización más eficiente del espectro y aumentando la capacidad de LTE-M [45] [46].

3.4. Tecnología NB-IoT

Esta tecnología radio se ha integrado en los estándares LTE, llegando a convivir con las demás tecnologías móviles. Se enfoca en facilitar una gran cobertura a despliegues de grandes cantidades de dispositivos, al menos 52547 dentro de una celda, con un ancho de banda mínimo de 180KHz (1 RB(*Resource Block*). Proporcionando dispositivos con una complejidad ultra baja, con un bajo coste y aportando una vida útil de batería de 10 años y al igual que LTE-M, una latencia para el 99 % de dispositivos de 10 segundos o menos [47] [48].

3.4.1. Arquitectura y pila de protocolos NB-IoT

Tanto su arquitectura como pila de protocolos derivan de LTE pero más reducida para conseguir que NB-IoT no tenga especificaciones que no vaya a utilizar y así no se sobrecargue.

La pila de protocolos incluye dos capas, la de usuario y la de control. En ambas capas se incluye 4 capas como se muestra en la Figura 3.10, si seguimos el modelo OSI, estas 4 capas se podrían reunir en únicamente 2 capas. La capa física, realiza la misma función que en LTE utilizando la multiplexación de SC-FDMA para el enlace ascendente. En cuanto a su capa de enlace, también es similar a la pila de protocolos de 4G 3.1.2 , utilizando los protocolos MAC, RLC y PDCP. Al plano de control se añaden dos protocolos, que se puede incluir en la capa de red, RRC también contenido en LTE y NAS se dedica a transferir la señal no radio entre la red troncal y el UE [46].

3.4.2. Espectro y Frecuencia NB-IoT

Al estar creado a partir de LTE, se puede implementar tres modos de operación para el uso del espectro de frecuencia [48] [49]:

- **Inband Mode:** Utiliza un ancho de banda específico (1 RB) dentro de una banda de frecuencias, por ejemplo de LTE. Es decir, comparte

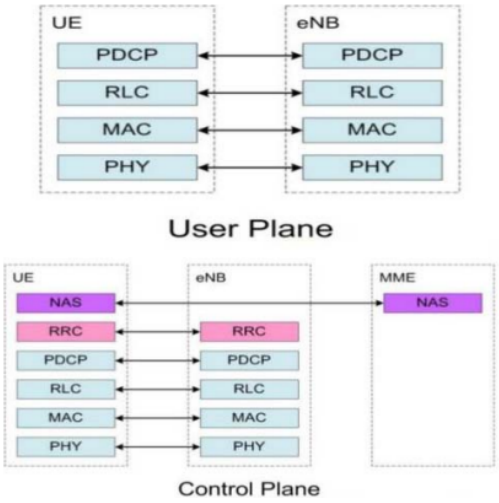

Figura 3.10: Pila de protocolos de NB-IoT [46]

el espectro con otra tecnología pero se usa otra modulación para que no haya interferencia.

- **Guardband Mode:** Las bandas de frecuencia utilizan una banda de protección llamada *guardband* en sus extremos. Este modo consiste en utilizar estas *guardband* como banda para NB-IoT.
- **Standalone Mode:** En los dos modos anteriores, se usaban bandas de LTE, este modo se despliega de forma independiente de LTE. Utilizando bandas exclusivas para NB-IoT, consiguiendo un mayor rendimiento y menos interferencia.

En la Figura 3.11 se tienen ejemplos de los tres modos para una mayor aclaración y un mejor compresión.

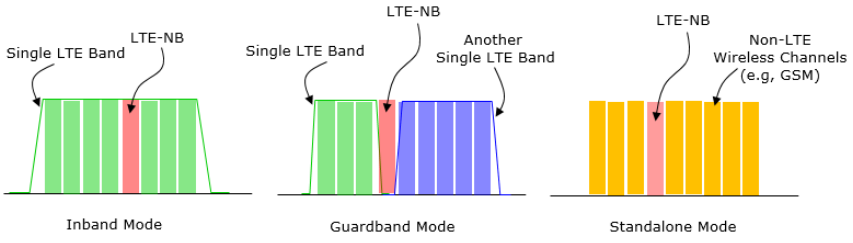

Figura 3.11: Modos de operación NB-IoT [49]

Al contrario de LTE-M, esta tecnología utiliza bandas de frecuencia definidas por 3GPP, en el *release 13* [5] se establecieron 14 bandas, posterior-

mente 4 más en el *release 14* [50] y 7 en el *release 15* [8],en la tabla 3.5 se muestran las bandas admitidas en Europa [51].

NB Band	**Uplink (MHz)**	**Downlink (MHz)**	**Duplex Mode**
B3	1710 – 1785	1805 – 1880	HD-FDD
B4	880 – 915	925 – 960	HD-FDD
B20	832 – 862	791 -821	HD-FDD

Tabla 3.5: Bandas de frecuencias de Nb-IoT [51]

Capítulo 4

Planificación y Coste

En este capítulo, se expondrá la planificación temporal a la hora de realizar las tareas, así como los recursos utilizados, hardware, software y humanos. Para finalizar, se incluirá un presupuesto total.

4.1. Planificación Temporal

En este primer apartado se muestra la planificación temporal mediante un diagrama de Gantt, véase en la Figura 4.1. A continuación, se indica el tiempo que se ha dedicado a cada uno de las tareas que se describen a continuación:

- **Estudio previo:** Estudios que se han realizado antes de la configuración y la ejecución de las pruebas. Consta del estudio de 4G, 5G, LTE-M y NB-IoT. Además del estudio de las diferentes alternativas que se encuentran actualmente en el mercado para desplegar estas tecnologías.

- **Redacción de la memoria:** En esta parte se redacta la memoria del TFG, con los resultados obtenidos. Tiene dos partes, la primera parte donde se escribe hasta el capítulo [3] de Fundamentos Teóricos. En la segunda parte se redacta toda la parte de configuración y pruebas junto a la sección de análisis de resultados. Además se incluye las conclusiones finales del proyecto con un apéndice que contiene algunos de los códigos utilizados.

- **Diseño y realización de pruebas:** En esta tarea se configuran las diferentes redes móviles (4G) y 5G) y redes IoT (LTE-M y NB-IoT) junto a sus archivos de configuración. Además del uso de estos para llevar a cabo las pruebas.

- **Análisis de resultados:** Se obtuvieron gráficas de los resultados mediantes códigos de Python para poder analizarlos y conseguir conclusiones.

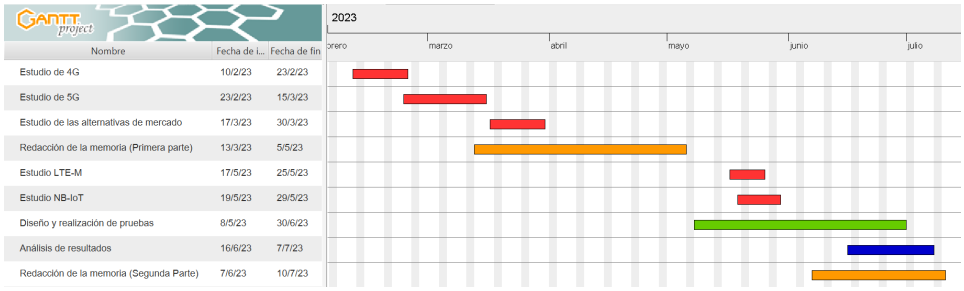

Figura 4.1: Diagrama de Gantt

A continuación se muestra, en la tabla 4.1, el desglose de las tareas con las horas empleadas en cada una de ellas.

Tareas	**Horas**
Estudio de 4G	20
Estudio de 5G	25
Estudio de las alternativas de mercado	30
Redacción de la memoria (Primera parte)	50
Estudio LTE-M	20
Estudio NB-IoT	20
Diseño y realización de pruebas	80
Análisis de resultados	30
Redacción de la memoria (Segunda Parte)	45
Total	320

Tabla 4.1: Horas empleadas

4.2. Recursos

En este apartado, se explicará que recursos se han empleado a lo largo del proyecto, recursos hardware, software y humanos

4.2.1. Recursos hardware

Los recursos hardware empleados se describen a continuación:

- **Ordenador Personal [52]:** Se utilizó un ordenador portátil personal, un ZenBook 13 UX325JA con un procesador i5-1035G1, una memoria

RAM de 16 GB y 1 TB de almacenamiento interno. Principalmente en la partes de investigación y documentación.

- **Servidor:** Ordenador de torre de gran rendimiento, ya que debe soportar el funcionamiento de nuestra red. Es utilizado como servidor LTE de *Amarisoft*.
- **Software Defined Radio (SDR)**: Las tarjetas SDR, han sido provistas por la empresa Amarisoft, se utiliza para desplegar el software de Amarisoft, consiguiendo con ello modificar los diferentes parámetros de la comunicaciones inalámbricas.
- **Antenas:** Se utilizan el modelo 5A2-0264-S01SP4-050 para obtener una conexión inalámbrica eficiente y confiable.
- **Modem Quectel [53]:** El modelo modem Quectel RM500Q-GL se emplea en este proyecto para establecer una conexión de datos en las distintas tecnologías móviles implementadas.
- **Tarjeta USIM:** Tarjeta empleada en el Modem quectel y en las motas Pycom. Se programó para almacenar información importante para la conexión.
- **Lector tarjetas USIM:** Se utiliza un lector proporcionado por la Universidad de Granada.
- **Jaula de Faraday:** Para obtener un medio libre de interferencias electromagnéticas, además para no afectar al entorno electromagnético con nuestra propia radiación, aislando los dispositivos usados en ella. En la Figura 4.2 se aprecia la jaula de Faraday utilizada, junto al Modem Quectel y las antenas en su interior.
- **Mota Pycom [54]:** Se trata del dispositivo utilizado para la conexión cuando se ha probado con la tecnología de NB-IoT y LTE-M.

4.2.2. Recursos software

Los recursos software que se utilizaron son:

- **Licencia Amarisoft, LTE 100 [23]:** Licencia privada para el uso del software de Amarisoft.
- **Visual Studio Code [55]:** Es un editor de código gratuito, que cuenta con una amplia gama de lenguajes de programación como Python o *Bash*. Se utiliza la versión 1.78.2.
- **MobaXterm [56]:** Programa gratuito para facilitar la conexión al servidor y a la quectel, mediante *Secure SHell* (SSH).

Figura 4.2: Quectel y antenas dentro de la jaula de Faraday

- **PySIM [57]:** Se trata de un software de código abierto que utiliza el lenguaje de programación Python para modificar los parámetros de la tarjetas *User Subscriber Identification Module* (USIM).
- **Wireshark [58]:** Es una herramienta que permite capturar y analizar el tráfico de una red, es gratuito y se usa la versión 3.6.7.
- **Iperf [59]:** Herramienta que permite realizar pruebas para medir el *throughput* entre dos dispositivos.
- **Oveleaf [60]:** Herramienta utiliza para redactar la memoria del proyecto utilizando LaTeX.

4.2.3. Recursos humanos

Con respecto a los recursos humanos se tendrá en cuenta el tiempo utilizado por cada una de las personas involucradas:

- **Jorge Navarro Ortiz:** Tutor del proyecto y profesor Titular de Universidad del Departamento de Teoría de la Señal, Telemática y Comunicaciones.
- **Félix Delgado Ferro:** Cotutor del proyecto e investigador predoctoral del Departamento de Teoría de la Señal, Telemática y Comunicaciones.
- **Javier Jiménez González:** Estudiante del grado de Ingeniería de Tecnologías de Telecomunicaciones.

Las horas empleadas en el desarrollo del trabajo de cada una de las personas descritas anteriormente se muestran en la tabla 4.2.

Persona	Total de horas
Jorge Navarro Ortiz	20
Félix Delgado Ferro	20
Javier Jiménez González	320

Tabla 4.2: Horas totales trabajadas

4.3. Presupuesto total

A la hora de realizar el presupuesto total, en primer lugar se calcularán el coste de cada uno de los recursos empleados. En la tabla 4.3 se observa el coste de los recursos hardware, en la tabla 4.4 los recursos software, para ambos apartados se calcula un tiempo de amortización de 3 años para cada elemento, por tanto, en la columna de coste por uso se calculará el precio total para los 6 meses de uso (3/36 precio total). Por último en la tabla 4.5 se muestran los recursos humanos junto al precio por hora de cada uno.

Recursos Hardware	Cantidad	Coste por unidad	Coste	Coste por uso
Ordenador	1	700,00 €	700,00 €	116,67 €
Servidor	1	2.000,00 €	2.000,00 €	333,33 €
SDR	2	1.500,00 €	3.000,00 €	500,00 €
Antenas	8	*0,00 €	0,00 €	0,00 €
Modem Quectel	1	340,00 €	340,00 €	56,67 €
Tarjetas USIM	3	10,00 €	30,00 €	5,00 €
Lector de tarjectas USIM	1	50,00 €	50,00 €	8,33 €
Jaula de Faraday	1	2.500,00 €	2.500,00 €	416,67 €
Mota Pycom	2	75,00 €	150,00 €	25,00 €
Coste Total			8.770,00 €	**1.461,67** €

Tabla 4.3: Coste recursos hardware

* Destacar que en el coste de las SDR incluye el precio de las antenas.

Recursos Software	Coste	Coste por uso
Licencia de Amarisoft	12000 €	2000 €
Visual Studio Code	0 €	0 €
MobaXterm	0 €	0 €
PySIM	0 €	0 €
Wireshark	0 €	0 €
Iperf3	0 €	0 €
Overleaf	0 €	0 €
Coste Total	12000 €	**2000** €

Tabla 4.4: Coste recursos software

Persona	Precio por hora	Total de horas	Coste
Jorge Navarro Ortiz	50,00 €	20	1.000,00 €
Félix Delgado Ferro	35,00 €	20	700,00 €
Javier Jiménez González	20,00 €	320	6.400 €
Coste Total	-	-	**8.100,00** €

Tabla 4.5: Coste recursos humanos

Recursos	Coste
Hardware	1.461,67 €
Software	2000 €
Humanos	8.100,00 €
Coste Total	**11561,67** €

Tabla 4.7: Presupuesto total

La estimación total del costo del proyecto sumando todos los recursos descritos anteriormente se recoge en la tabla 4.7, obteniendo un coste total del proyecto de **11.561,67** €.

Capítulo 5

Configuración y Pruebas

En este apartado se dispondrá a dar una explicación de la implementación realizada en el proyecto. Para ello, se expondrán los pasos seguidos junto a los recursos usados y su configuración. Se describirá la implementación de la red 4G LTE, 5G SA, 5G NSA, además de las tecnologías IoT LTE-M y NB-IoT. Por último, se explicarán las pruebas realizadas para cada una de las tecnologías desplegadas.

5.1. Instalación y configuración del software de Amarisoft

En la realización del proyecto se ha utilizado el software privado Amarisoft que cuenta con herramientas propias para comprobar el correcto funcionamiento del despliegue, en el manual oficial [61] se encuentra toda la información sobre estas herramientas.

Utilizando el comando *screen -x lte* se puede acceder a los componentes eNB, MME, IMS o MBMSGW para obtener datos sobre las celda y los usuarios conectados, como se observa en la Figura 5.1. Se puede identificar como al MME se ha conectado el usuario con el *Subscription Permanent Identifier* (SUPI) 901700000000002, número que coincide con el IMSI, su dirección IP es "192.168.3.2". En el apartado de eNB se muestra al usuario conectado a la estación base en este caso su identificador es 901700000000002. También se observa información de la celda, como el PLMN o la banda y la frecuencia que utilizan cada una de las tecnologías móviles que se despliegan.

Otra herramienta por la que se ha elegido este software es la interfaz web que nos ofrece Amarisoft que ayuda a poder analizar los mensajes generados en la red. A esta GUI (*Graphical User Interface*) se puede acceder mediante un navegador web introduciendo *o http://[IPserver]/lte*.

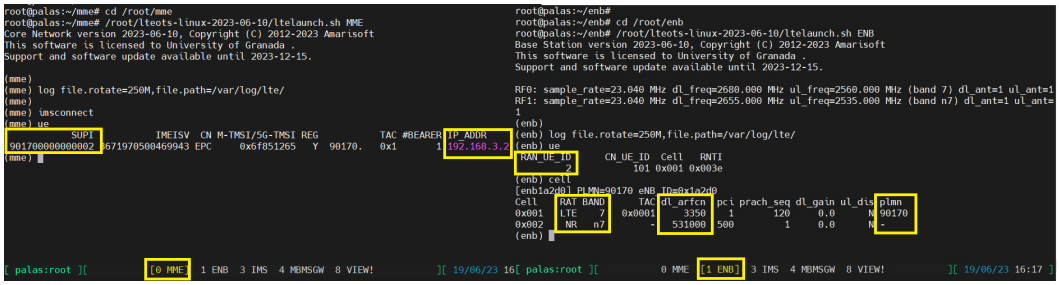

Figura 5.1: Ejemplo del comando *screen -x lte*

A la hora de realizar la instalación de Amarisoft se ha seguido la guía de instalación oficial del software LTE 100 [25], cumpliendo en todo momento con los requerimientos dados. Se utilizará como servidor un ordenador de propósito general con el sistema operativo de *Ubuntu 18.04* que cuenta con unas altas características en rendimiento como indican las especificaciones dadas por Amarisoft.

Antes de empezar con la instalación del software, se debe conectar las tarjetas SDR proporcionadas por Amarisoft. En nuestro caso se cuenta con dos tarjetas que se deben conectar desde el conector *OUT* de la primera tarjeta al conector *IN* de la segunda, tal y como se muestra en la Figura 5.2. Posteriormente se instalaron en nuestro servidor.

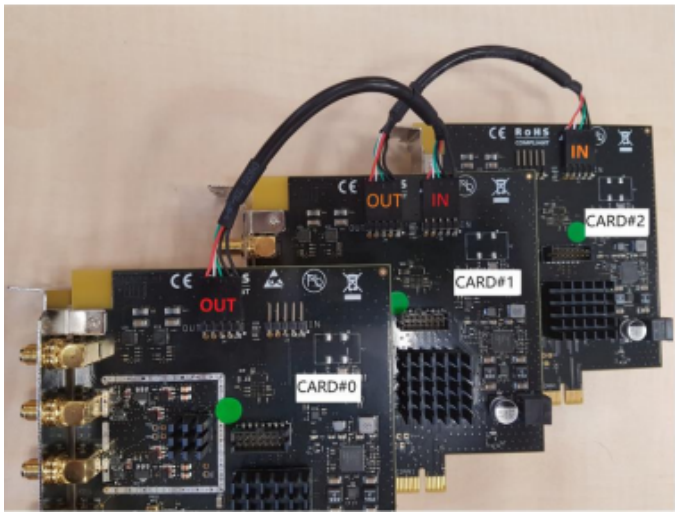

Figura 5.2: Conexión entre SDRs

Tras instalar el hardware necesario se procede a la instalación del software. Se debe descargar de la página oficial de Amarisoft [23] los archivos

ejecutables y binarios para posteriormente extraerlos en el ordenador. La carpeta tendrá la fecha del *release* en el que está el software. En este directorio, se encuentra el instalador *install.sh* con el que se puede instalar automáticamente los diferentes nodos. Se ejecutará como *root* mediante el comando *./install.sh path -default*, donde *path* indica la ruta de instalación de los componentes. Si se deja en blanco su instalación se realiza en */root*, en nuestro caso se llevó a cabo sobre el directorio */opt*. La opción *default* fuerza la respuesta predeterminada de las preguntas que se realizan durante la instalación, si se desea una instalación personalizada, se debe ejecutar sin esta opción y responder cada pregunta por separado. Si la instalación es correcta, por pantalla aparece el mensaje *Installation succesful* junto a la respuesta predeterminada, como se muestra en la Figura 5.3.

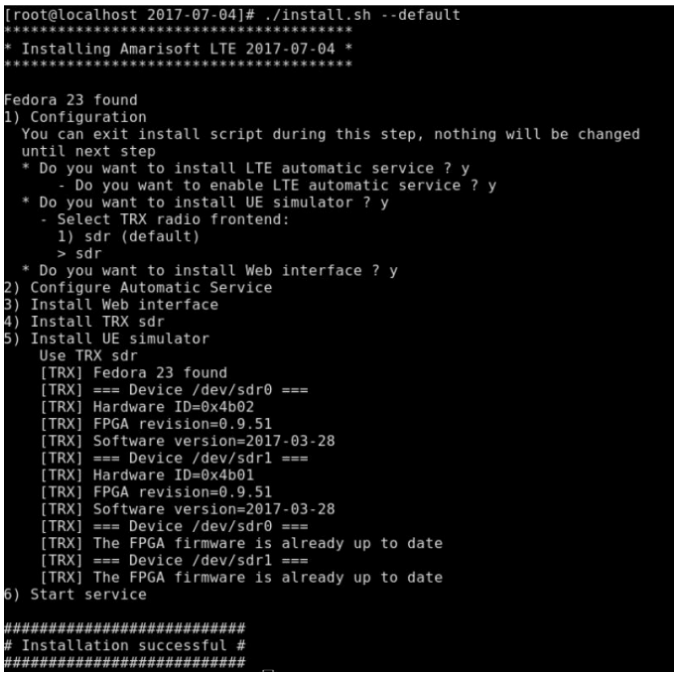

Figura 5.3: Respuesta predeterminada para LTE 100 UE

Después de la instalación, se necesitará los archivos de la licencia para la activación del sistema. Amarisoft cuenta con tres tipos de licencia, *fixed license*, *floating license with license server* y *floating license in USB dongle*, en nuestro caso se utilizó la última. Para obtener la licencia, únicamente se deberá conectar el *Universal Serial Bus* (USB) al nuestro ordenador de uso genérico que funciona como servidor. Si el USB, no se monta de forma automática se puede hacer manualmente, en primer lugar, sino detecta el USB se debe escribir *fdisk -l*, posteriormente crear una carpeta para el montaje y ejecutar el comando *mount source=devicename directory*, para simplificar

estos pasos se creó un *script*, con estos comandos.

Como se cuenta con dos tarjetas SDR se deben configurar los drivers del controlador *Radio Frequency* (RF), para ello, se debe identificar el nombre de cada SDR y su número único utilizando el comando *sdr mapping* para saber cuál es la maestro y cuál es la tarjeta esclavo. Posteriormente, es necesario editar el archivo del mapeo con el orden obtenido.

5.1.1. Despliegue de 4G y 5G

Tras instalar el software se debe comprobar los archivos de configuración de los diferentes módulos y ver qué tipo de configuración se desea desplegar. Nuestra red se separa en la parte de acceso formada por el componente de eNB, que toma funcionalidad de gNB y por la parte troncal compuesta por el MME y el *IP Multimedia Subsystem* (IMS). A continuación, se explicará la función de cada uno de estos componentes junto a los principales parámetros que pueden ser configurables dependiendo del despliegue.

Configuración del módulo MME

El módulo de MME, gestiona la movilidad, el establecimiento y control de acceso junto a la seguridad de la red. Su archivo de configuración es el *mme.cfg* y se encuentra en el directorio *opt/mme/config*.

En la Tabla 5.1 se indican los principales parámetros configurables junto a una pequeña descripción. Estos valores son utilizados a lo largo del desarrollo del proyecto para todos los despliegues.

Configuración del módulo IMS

El módulo de IMS permite la funcionalidad de los servicios multimedia basados en IP sobre la red. Se necesita para la configuración de los datos de los UE. Cuenta con dos archivos de configuración *ims.cfg* y una base de datos para la autenticación y autorización de los usuarios *ue_db-ims.cfg*, se encuentran en *opt/mme/config*.

En la Tabla 5.2 se describe los principales parámetros del archivo de configuración *ims.cfg*. Los datos de mayor importancia del fichero *ue_db-ims.cfg* se pueden ver en la Tabla 5.3, donde se destaca que el valor indicado es el del primer usuario, a lo largo del proyecto se utilizaron tres usuarios que siguieron la misma estructura, como se muestra en la Figura 5.4. En cuanto al *IP Multimedia Private Identity* (IMPI), está formado por *imsi@domain* y el IMPU por el IMSI y el número de teléfono. Al igual que en el MME

Parámetro	Descripción	Valor
com_addr	Puerto para comunicarse con el MME	0.0.0.0:9000
gtp_addr	Dirección IP para GTP-U	127.0.1.100
plmn	Identificación de área y operador	90170
mme_group_id	Identificador de grupo MME	32769
emergency_number_list	Lista con número de emergencia	911, 112
network_name	Identificador de la red con un nombre	Amarisoft Network
network_short_name	Identificador de la red con un nombre corto	Amarisoft
pdn_list	Listado de las redes de datos de paquetes	-
pdn_type	Indica el protocolo usado	ipv4
access_point_name	Identifica el punto de acceso	internet
first_ip_addr	Primera IP que asigna el MME	192.168.3.2
last_ip_addr	Última IP que asigna el MME	192.168.3.254
dns_addr	Especifica IP del DNS	8.8.8.8

Tabla 5.1: Principales parámetros de configuración de MME

estas variables no varían para todos los despliegues. Además, esta configuración debe ser la misma que en la USIM para que el proceso de registro y posteriormente de autenticación sean correctos.

Parámetro	Descripción	Valor
com_addr	Puerto para comunicarse con el IMS	0.0.0.0:9003
domain	Nombre de dominio global	amarisoft.com
include	Nombre de base de datos de usuario	ue_db-ims.cfg

Tabla 5.2: Principales parámetros de configuración de IMS

Parámetro	Descripción	Valor
multi_sim	Opción para permitir varios usuarios	true
authent_type	Tipo de autenticación	MD5
sim_algo	Algoritmo de autenticación	milenage
imsi	Código de identificación usuario	901700000000001
opc	Clave para algoritmo de autenticación	000102030405060708090 A0B0C0D0E0F
amf	Campo de gestión de autenticación	0x9001
sqn	Secuencia empleado en autenticación	000000000000
K	Clave utilizada entre algoritmo y red	00112233445566778899 AABBCCDDEEFF
impi	Identificador privado del usuario en la red	901700000000001@ims.mnc070. mcc901.3gppnetwork.org
impu	Identificador público del usuario en la red	901700000000001, tel : 600000001
domain	Identificador del dominio de la red	ims.mnc070.mcc901.3gppnetwork.org

Tabla 5.3: Parámetros de configuración de la base de datos de IMS

```
// Configure USIM PLMN 90170
    sim_algo: "milenage",
    imsi: "901700000000003",
    opc: "000102030405060708090A0B0C0D0E0F",
    amf: 0x9001,
    sqn: "000000000000",
    K: "00112233445566778899AABBCCDDEEFF",
   impi: "901700000000003@ims.mnc070.mcc901.3gppnetwork.org",
    impu: ["901700000000003", "tel:0600000003"],
        domain: "ims.mnc070.mcc901.3gppnetwork.org",
}, {
```

Figura 5.4: Configuración del usuario 3 en la base de datos de IMS

Configuración del módulo eNB

Este módulo hace de estación base de la red, sus principales responsabilidades son la transmisión y recepción de la señal a las SDR, el control del acceso a los dispositivos y de la interconexión entre el UE y el EPC. Su archivo de configuración es *enb.cfg* y se encuentra en la ruta *opt/enb/config*. Al contrario de los módulos anteriores (MME, IMS), el eNB utiliza una configuración diferente para cada despliegue. A continuación, para 4G y 5G se describen los parámetros configurables más relevantes junto a unas variables predefinidas que ayudan a cambiar la configuración de la estación base.

- **Despliegue de 4G:** Se emplea el archivo por defecto *enb.cfg* y en la Tabla 5.4 se muestran algunos parámetros de configuración, el valor indicado es el utilizado a la hora de realizar las pruebas. Tanto las variable *mme_addr* como *gtp_addr* se deben cambiar si el MME se encuentra en un *host* diferente, en la Figura 5.5 se indica esta explicación que ha sido dada por Amarisoft en el fichero de configuración. Cabe destacar que el valor de *dl_earfcn* depende del tipo de duplexación y la banda de frecuencia empleados. En el caso de TDD se utiliza el *Absolute Radio Frequency Channel Number* (ARFCN) 40620, un identificar asignado a cada canal radio, con el se puede calcular la frecuencia exacta del canal. La frecuencia central de portadora para *downlink* de 2593 MHz, es decir, la banda 41 como se indica en la sección 3.1.3. Una frecuencia central de 2680 MHz (banda 7) y el canal 3350 se emplea en FDD.

  En la Tabla 5.5 se describen las variables con todas las opciones que se han modificado para las diferentes pruebas realizadas para la tecnología móvil de 4G.

- **Despliegue de 5G-NSA:** Se utiliza el archivo de configuración *gnb-nsa.cfg*, donde se despliegan tanto celdas para LTE como para NSA. Por tanto, sus parámetros de configuración son similares a los de 4G,

```
mme_list: [
  {
    /* address of MME for S1AP connection. Must be modified if the MME
       runs on a different host. */
    mme_addr: "127.0.1.100",
  },
],
/* GTP bind address (=address of the ethernet interface connected to
   the MME). Must be modified if the MME runs on a different host. */
gtp_addr: "127.0.1.1",

/* high 20 bits of SIB1.cellIdentifier */
enb_id: 0x1A2D0,

/* list of cells */
cell_list: [
{
  /* Broadcasted PLMN identities */
  plmn_list: [
    "90170",
  ],
```

Figura 5.5: Configuración del eNB para 4G.

Parámetro	Descripción	Valor
mme_addr	Dirección IP para la conexión al MME con S1AP	127.0.1.100
gtp_addr	Dirección enlace GTP para la comunicación del MME	127.0.1.1
cell_list	Lista de celdas que se quieren crear	-
plmn_list	Lista de los PLMN transmitidas en difusión	90170
dl_earfcn	Frecuencia de la portadora para DL	40620, 2593 MHz (Band 41) 3350, 2680 MHz (Band 7)
n_id_cell	Identificador de la celda física	1
cell_id	Identificador de celda en forma hexadecimal	0x01
tac	Identificador del tipo de área de celda	0x0001

Tabla 5.4: Principales parámetros de configuración de 4G

Variable	Descripción	Valores
TDD	Tipo de duplexación	0 (FDD), 1 (TDD)
N_RB_DL	Ancho de banda de la transmisión	6 (1.4 MHz), 15 (3 MHz), 50 (10 MHz), 75 (15 MHz), 25 (5 MHz), 100 (20 MHz)
N_ANTENNA_DL	Número de antenas usadas en DownLink	0 (SISO), 2 (MIMO 2x2)
N_ANTENNA_UL	Número de antenas usadas en UpLink	1 (SISO), 2 (MIMO 2x2)
CHANNEL_SIM	Simulador de canal	0 (disabled), 1 (enabled)

Tabla 5.5: Variables de configuración de 4G

solamente se añaden nuevos parámetros para la celda de NR, en la Tabla 5.6 se muestra los principales parámetros utilizados en este archivo.

Parámetro	Descripción	Valores
amf_addr	Dirección IP para la conexión al AMF con NGAP	127.0.1.100
gtp_addr	Dirección enlace GTP para comunicación del AMF	127.0.1.1
nr_cell_list	Lista de las celdas que se utilizarán	-
cell_id	Identificador de celda en forma hexadecimal	0x01
band	Banda de frecuencia utilizada	257 41 7
dl_nr_arfcn	Frecuencia de la portadora para DL	2079167 (28000.08 MHz) 518601 (2593 MHz) 531000 (2655 MHz)
subcarrier_spacing	Espaciado entre subportadora	120 kHz 30 kHz 15 kHz
n_id_cell	Identificador de la celda física	500
tac	Identificador del tipo de área de celda	100
plmn	Identificación de área y operador	90170

Tabla 5.6: Principales parámetros de configuración de 5G

En cuanto a sus variables de configuración, se añaden nuevas y se definen en la Tabla 5.7.

Variable	Descripción	Valores
TDD	Tipo de duplexación de LTE	0(FDD) 1(TDD)
NR_TDD	Tipo de duplexación de NR	0(NR FDD) 1(NR TDD)
FR2	Banda de frecuencia de la transmisión	0 (FR1) 1 (FR2)
NR_TDD_CONFIG	Patrones de configuración de NR para TDD	FR1: 1, 2, 3, 4 FR2: 10
N_RB_DL	Ancho de banda de la transmisión	6 (1.4 MHz), 15 (3 MHz) 50 (10 MHz), 75 (15 MHz) 25 (5 MHz), 100 (20 MHz)
N_ANTENNA_DL	Número de antenas usadas en DownLink	0 (SISO), 2 (MIMO 2x2)
TRX_MAX_BANDWIDTH	Máximo ancho de banda que soporta una SDR	50 (50MHz), 100 (100MHz)
NR_BANDWITH	Ancho de banda de la transmisión	10 MHz, 20 MHz 30 MHz, 40 MH 50 MHz
USE_SRS	Señal de Referencia de Sondeo	0(Disabled), 1(Enabled)

Tabla 5.7: Variables de configuración de 5G-NSA

- **Despliegue de 5G-SA:** En el proceso de configuración de esta tecnología se ha utilizado el fichero *gnb-sa.cfg*, los principales parámetros

de configuración son los mismos que los mostrados para 5G-NSA y se definen en la Tabla 5.6. En la Figura 5.6 se muestra la configuración de la celdas, dependiendo de su duplexación (TDD/FDD) y su rango de frecuencia (FR1 y FR2).

```
  /* list of cells */
  cell_list: [],

  nr_cell_list: [
  {
    rf_port: 0,
    cell_id: 0x01,
#if NR_TDD == 1
#if FR2
    band: 257,
    dl_nr_arfcn: 2079167,  /* 28000.08 MHz */
    subcarrier_spacing: 120, /* kHz */
    ssb_pos_bitmap: "01000000000000000000000000000000000000000000000000000000000000000000000000000000",
#else
    band: 78,
    dl_nr_arfcn: 632628,  /* 3489.42 MHz */
    subcarrier_spacing: 30, /* kHz */
    ssb_pos_bitmap: "10000000",
#endif
#else
    band: 7,
    dl_nr_arfcn: 536020,  /* 2680 MHz */
    ssb_subcarrier_spacing: 15,
    subcarrier_spacing: 15, /* kHz */
    ssb_pos_bitmap: "1000",
#endif
```

Figura 5.6: Configuración del eNB para 5G-SA.

Las variables que se han ido modificando para la realización de las distintas pruebas dando lugar a diferentes configuraciones de la red, aparecen en la Tabla 5.8.

Variable	Descripción	Valores
NR_TDD	Tipo de duplexación	0(NR FDD) 1(NR TDD)
FR2	Banda de frecuencia de la transmisión	0 (FR1) 1 (FR2)
NR_TDD_CONFIG	Patrones de configuración de NR para TDD	FR1: 1, 2, 3, 4 FR2: 10
N_ANTENNA_DL	Número de antenas usadas en DownLink	1 (SISO) 2 (MIMO 2x2)
N_ANTENNA_UL	Número de antenas usadas en UpLink	1 (SISO) 2 (MIMO 2x2)
NR_BANDWITH	Ancho de banda de la transmisión	10 MHz, 20 MHz, 30 MHz, 40 MHz 50 MHz

Tabla 5.8: Variables de configuración de 5G

Cabe destacar que los cambios en las variables se pueden realizar di-

rectamente en su respectivo archivo de configuración o realizar un enlace simbólico con el comando ***ln -s ficheroconf enb.cfg*** para apuntar al fichero con la configuración deseada. Una vez realizado, se puede comprobar si se ha ejecutado correctamente con el comando *ls -la* para mostrar una lista de todos los archivos incluyendo los ocultos. En la Figura 5.7 se muestra un ejemplo de este enlace simbólico. Tras cambiar la configuración, se debe reiniciar el servicio con el comando ***service lte restart*** o pararlo ***service lte stop*** y volverlo a lanzar ***service lte start*** si el servicio no estaba activo antes del cambio de configuración.

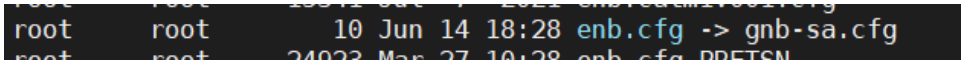

Figura 5.7: Ejemplo de enlace simbólico.

5.1.2. Despliegue de LTE-M y NB-IoT

En esta sección se van a describir las configuraciones de la estación base para las tecnologías IoT de LTE-M y NB-IoT, es decir el archivo *enb.cfg*. En cuanto a la configuración de la parte troncal, MME e IMS es igual a la descrita en la sección 5.1.1.

- **Despliegue de LTE-M:** Se trabajó sobre el fichero *enb-catm1.cfg* dado por Amarisoft. Antes de realizar el nuevo enlace simbólico con este fichero se debe eliminar el archivo *enb.cfg*. En este nuevo archivo se despliega una red LTE y CAT-M1, gracias a que este último se basa en LTE. Por tanto, las variables de configuración que se cambian para la realización de las pruebas son las mismas que las descritas en la Tabla 5.5, de igual manera ocurre con los principales parámetros que aparecen en la Tabla 5.4.

  Ya que los principales parámetros y variables del despliegue de 4G son similares a las del despliegue LTE-M, se procede a explicar en la Figura 5.8 los cambios realizado en *enb-catm1.cfg* con respecto al archivo de configuración de 4G, donde ***cqi_pucch_n_rb*** se define con el valor 2, ya que se pueden conectar dos usuarios, uno para LTE y otro para LTE-M. Además se añade la configuración para LTE-M.

- **Despliegue de NB-IoT:** Se usó el fichero *enb-nbiot.cfg*, al igual que en el caso anterior se debe eliminar el fichero del enlace simbólico anterior si así se requiere y crear el nuevo enlace simbólico. Se despliega NB-IoT y la red LTE. En este caso las variables de configuración son distintas cómo se muestra en la Tabla 5.9, destacar *NB_MODE*, donde

```
    /* PUCCH dedicated config (currently same for all UEs) */
    pucch_dedicated: {
      n1_pucch_sr_count: 11, /* increase if more UEs are needed */
      cqi_pucch_n_rb: 2, /* increase if more UEs are needed */
#if TDD == 1
      //tdd_ack_nack_feedback_mode: "bundling", /* TDD only */
      tdd_ack_nack_feedback_mode: "multiplexing", /* TDD only */
#endif
    },

    /* PUSCH dedicated config (currently same for all UEs) */
    pusch_dedicated: {
      beta_offset_ack_index: 9,
      beta_offset_ri_index: 6,
      beta_offset_cqi_index: 6,
    },
```

Figura 5.8: Cambios realizados en archivo *enb-catm1.cfg*

se utilizará el modo de operación 0, que significa que las dos tecnologías operan en la misma frecuencia y *LTE_DL_EARFCN* que indica el canal utilizado. Además se añade la configuración para la celda de NB-IoT como se muestra en la Figura 5.9.

Parámetro	Descripción	Valor
NB_MODE	Modo de operación	0 (In Band) 1 (Guard Band) 2 (Standalone)
NB_NON_ANCHOR	Utilización una portadora no principal	0 (non anchor carrier disabled) 1 (non anchor carrier enabled)
LTE_DL_EARFCN	Canal utilizado	3350 (FDD - 2680 MHz [Band 7])
N_RB_DL	Ancho de banda de la transmisión	25 (5MHz), 50 (10MHz) 75 (15MHz), 100 (20MHz)
N_ANTENNA_DL	Número de antenas usadas en DownLink	1 (SISO) 2 (MIMO 2x2)
N_ANTENNA_UL	Número de antenas usadas en UpLink	1, 2
CHANNEL_SIM	Simulador de canal	0 (channel simulator disabled) 1 (channel simulator enabled)

Tabla 5.9: Variables de configuración de NB-IoT

5.1.3. Configuración de los dispositivos

Posteriormente de la configuración de los componentes de la red, se debe configurar los dispositivos que se conectará a la red. Para 4G y 5G se utilizará un modem quectel, para LTE-M y NB-IoT una mota pycom. Destacar que en cada uno de los dispositivos contamos con una tarjeta USIM que ha sido configurada.

```
/* list of NB-IoT cells */
nb_cell_list: [
  {
    plmn_list: [
      { plmn: "90170", reserved: false },
    ],
    cell_id: 0x02,
    tac: 0x0002, /* SIB1.trackingAreaCode */
    n_antenna_dl: 1,
    n_antenna_ul: 1,
#if NB_MODE == 0
    operation_mode: "same_pci",
#if N_RB_DL == 25
    dl_prb: 17, /* DL PRB number in the base LTE cell */
    ul_prb: 17, /* UL PRB number in the base LTE cell */
#elif N_RB_DL == 50
    dl_prb: 19, /* DL PRB number in the base LTE cell */
    ul_prb: 19, /* UL PRB number in the base LTE cell */
#elif N_RB_DL == 75
    dl_prb: 17, /* DL PRB number in the base LTE cell */
    ul_prb: 17, /* UL PRB number in the base LTE cell */
#elif N_RB_DL == 100
    dl_prb: 19, /* DL PRB number in the base LTE cell */
    ul_prb: 19, /* UL PRB number in the base LTE cell */
#endif
    nrs_crs_power_offset: 0, /* in dB (same_pci only) */
    base_cell_id:0x01,
```

Figura 5.9: Configuración celda para NB-IoT

Configuración tarjetas USIM

Estas tarjetas tienen información de gran importancia y personalizada para autenticar e identificar a los usuarios de la red. Como nuestra estación base no pertenece a ningún operador, se debe programar para cada usuario una tarjeta USIM, para ello se utilizará la herramienta *pysim* [57] junto a un lector especial, para poder acceder a la tarjeta a través de nuestro ordenador.

Los principales parámetros almacenados en la USIM son:

- **Integrated Circuit Card Identifier (ICCID) [62]:** En nuestro caso contamos con 3 tarjetas, para diferenciarlas se emplea este número. Es un identificador único de cada tarjeta USIM de hasta 20 dígitos, en la Figura5.10 se muestra una con varios ejemplos con el significado de cada número que compone este parámetro. Se encuentra impreso en la misma tarjeta o se puede obtener mediante el comando ***./pySim-read.py -p0***.
- **International Mobile Subscriber Identify (IMSI) [63]:** Para identificar a un suscriptor en la red y poder diferenciarlos de los demás usuarios, está formado por 15 dígitos, donde los 3 primeros identifican el país de emisión, los 2 o 3 siguientes la red dentro del país indicado

	ICCID								
BSNL	**89915590631419586640**								
	Industry Identifier	**Country Code**	**Issuer Identifier**	**Month and year of manufacturing**	**Configuration code**	**SIM number**	**Checksum digit**	**Issuer Identification Number**	**Individual Account Identification Number**
	89	91	559	0613	41	958664	0	8991559	063141958664
Idea	**89918951031446662125**								
	Industry Identifier	**Country Code**	**Issuer Identifier**	**Month and year of manufacturing**	**Configuration code**	**SIM number**	**Checksum digit**	**Issuer Identification Number**	**Individual Account Identification Number**
	89	91	895	1031	44	666212	5	8991895	103144666212
Voda-fone	**89911500021855394196**								
	Industry Identifier	**Country Code**	**Issuer Identifier**	**Month and year of manufacturing**	**Configuration code**	**SIM number**	**Checksum digit**	**Issuer Identification Number**	**Individual Account Identification Number**
	89	91	150	0021	85	539479	6	8991150	002185539479

Figura 5.10: Interpretación ICCID con detalle[62]

y los 9 o 10 últimos dígitos se trata del número de identificación de suscriptor dentro de la red.

- **Authentication key (Ki):** Se trata de la clave utilizada en la comunicación:
- **Location Area Identity (LAI):** Utilizado para identificar el área donde se encuentra el usuario.
- **Administrative Key (ADM1):** Es una clave administrativa para poder administrar la configuración de la tarjeta SIM.

Para la configuración solamente se empleará el ICCID para identificar la tarjeta y poder establecerle el IMSI deseado. Se utilizará un *script* (Anexo B con los ICCID de las tarjetas con las que contamos y la asignación del IMSI para facilitar el trabajo . Destacar que esta configuración debe ser la misma que la realizada anteriormente en el IMS.

Configuración modem quectel

En las tecnologías móviles de 4G y 5G se utilizó un modem quectel con una de las tarjetas USIM. Los pasos para su instalación se muestran a continuación:

```
# Assuming we are in the $HOME/quectel directory
unzip QConnectManager_Linux_V1.6.1.zip
cd quectel-CM
make
sudo cp quectel-CM /usr/bin
```

```
git clone git://busybox.net/busybox.git
cd busybox
git checkout remotes/origin/1_35_stable
cd ..
sudo mkdir /usr/share/udhcpc
sudo cp busybox/examples/udhcp/simple.script /usr/share/udhcpc/default.script
sudo chmod 755 /usr/share/udhcpc/default.script
```

A continuación, se despliega una explicación más detallada de cada paso. En primer lugar, se debe descomprimir el archivo *ConnectManager_Linux_V1.6.1.zip*, descargado previamente de la página oficial de Quectel [53], con ello se crea el directorio *quectel-CM*. Se debe acceder a este directorio y emplear el comando *make* para compilarlo y crear el ejecutable que se debe copar a */usr/bin*. Además, se utiliza la herramienta *BusyBox* [64], que junta varias herramientas de línea de comandos en un solo ejecutable, muy utilizado en sistemas con limitaciones de recursos. Se clona el repositorio de *BusyBox* y se debe copiar el *script simple.script* a una nueva carpeta creada en */usr/share/udhcpc*.

Posteriormente se debe ejecutar el comando ***sudo ./quectel-CM &*** para comprobar si su instalación fue correcta y ver la dirección IP del dispositivo junto a su nombre y la puerta de enlace. El servicio por parte de Amarisoft debe estar lanzado para que se pueda realizar la conexión. Para simplificarlo se realizó un *script* (Anexo A) para lanzarlo desde la quectel y obtener una conexión como se observa en la Figura 5.11. Donde en primer lugar se ve como la quectel intenta realizar la conexión, detectando finalmente la configuración (5G-NSA), también se ve el nombre del dispositivo y por último su dirección IP (192.168.3.2) y la puerta de enlace (192.168.3.1), dirección que había sido modelada en los archivos de configuración.

Algo que se debe tener en cuenta al realizar la conexión con la quectel es la banda de frecuencia que se utiliza ya que dependiendo del espectro de frecuencia se puede admitir un espaciado de portadora y un ancho de banda distinto. También destacar que nuestra tarjeta Quectel tiene una limitación con 5G-SA ya que solo soporta un *subcarrier spacing* de 15 kHz para FDD, en cambio para TDD consigue un valor hasta 20 kHz. Se puede revisar su banda y ancho de banda mediante un mensaje *UE capability information* que envía el UE al conectarse a la red. Para nuestro modelo las bandas aceptadas para 4G y 5G se muestran en las tablas 5.10 y 5.11 respectivamente:

Configuración mota pycom

Con las tecnologías de LTE-M y NB-IoT se emplea la mota pycom con una USIM. En este caso, se utilizaron dos motas con dos tarjetas diferentes, una para cada tecnología. Para poder configurar la mota, se debe conectar

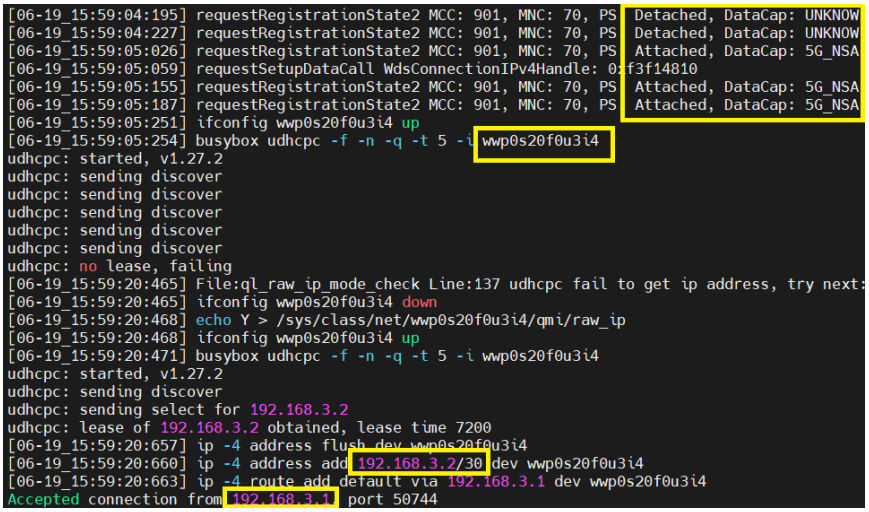

Figura 5.11: Ejemplo de conexión de tarjeta Quectel

Banda	Ancho de banda admitido
B1	1.4, 3, 5, 10, 15, 20 MHz
B2	1.4, 3, 5, 10, 15, 20 MHz
B3	1.4, 3, 5, 10, 15, 20 MHz
B4	1.4, 3, 5, 10, 15, 20 MHz
B5	1.4, 3, 5, 10, 15, 20 MHz
B7	1.4, 3, 5, 10, 15, 20 MHz
B8	1.4, 3, 5, 10, 15, 20 MHz
B12	1.4, 3, 5, 10 MHz
B13	1.4, 3, 5, 10 MHz
B17	1.4, 3, 5 MHz
B18	1.4, 3, 5 MHz
B19	1.4, 3, 5 MHz
B20	1.4, 3, 5, 10 MHz
B25	1.4, 3, 5 MHz
B26	1.4, 3, 5 MHz
B28	1.4, 3, 5 MHz
B29	1.4, 3, 5 MHz
B30	1.4, 3, 5 MHz
B38	5, 10, 15, 20 MHz
B39	5, 10, 15, 20 MHz
B40	5, 10, 15, 20 MHz
B41	5, 10, 15, 20 MHz
B66	1.4, 3, 5, 10 MHz

Tabla 5.10: Banda y ancho de banda admitidos en 4G para el modem Quectel RM500Q-GL

Banda	Espaciado de subportadora (SCS)	Ancho de banda admitido
n79	30kHz	40, 50, 60, 80 MHz
n78	30kHz	20, 30, 40, 50, 60, 80 MHz
n77	30kHz	20, 30, 40, 50, 60, 80 MHz
n71	15kHz	5, 10, 15, 20 MHz
n66	15kHz	5, 10, 15, 20, 30, 40 MHz
n48	30kHz	10, 20, 40, 50, 60, 80 MHz
n41	30kHz	10, 20, 40, 50, 60, 80 MHz
n40	30kHz	10, 20, 40, 50, 60, 80 MHz
n38	-	20, 30, 40 MHz
n28	15kHz	5, 10, 15, 20, 30 MHz
n25	15kHz	5, 10, 15, 20 MHz
n20	15kHz	5, 10, 15, 20 MHz
n12	15kHz	5, 10, 15 MHz
n8	15kHz	5, 10, 15, 20 MHz
n7	15kHz	5, 10, 15, 20 MHz
n5	15kHz	5, 10, 15, 20 MHz
n3	15kHz	5, 10, 15, 20, 25, 30 MHz
n2	15kHz	5, 10, 15, 20 MHz
n1	15kHz	5, 10, 15, 20, 30, 40 MHz

Tabla 5.11: Banda, espaciado de subportadora y ancho de banda admitidos en 5G para el modem Quectel RM500Q-GL

al ordenador mediante un cable USB y se utilizó la herramienta *Visual Studio Code* y su extensión *Pymakr* para entrar mediante una terminal en la mota Pycom, además de escribir y cargar código en el lenguaje micropython.

En primer lugar se debe revisar que *firmware* tiene instalado, a través de *Pymakr* se empleó la terminal para descargar la librería *sqnsupgrade* y poder utilizar el comando ***sqnsupgrade.info()*** para ver que el *firmware* actual de la pycom. En la Figura 5.12 se ve la información mostrada tras ejecutar el comando anterior, hay dos tipos de *firmware*, LR5.xx para CAT-M1 y LR6.xx para NB-IoT. Si tras realizar estos pasos, el *firmware* no es el deseado se deberá actualizar, para ello se descargará el fichero en una tarjeta SD y se puede montar con el siguiente código:

```
from machine import SD

sd = SD()
os.mount(sd, '/sd')    # mount it
os.listdir('/sd')      # list its content
```

Para actualizar se deberá ejecutar el comando y resetear la Pycom:

```
>>> import sqnsupgrade
>>> sqnsupgrade.run('/sd/nombre_archivo', '/sd/updater.elf')
```

```
>>> sqnsupgrade.info()
<<< Welcome to the SQN3330 firmware updater [1.2.6] >>>
>>> FiPy with firmware version 1.20.2.r6
Your modem is in application mode. Here is the current version:
UE6.0.0.0
LR6.0.0.0-41019

IMEI: 354346096998918
```

Figura 5.12: Ejemplo de revisión firmware de la mota pycom

Al igual que con la tarjeta quectel, la mota pycom solo soporta algunas bandas de LTE, estas bandas se pueden ver en el mensaje *UE capability information*. Las bandas aceptadas por nuestra mota son: 1, 2, 3, 4, 5, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28, 66, como se muestra en la Figura 5.13.

Figura 5.13: Bandas LTE permitidas por la mota Pycom

5.2. Pruebas realizadas

En este apartado se van a describir las pruebas realizadas para las diferentes tecnologías móviles tras llevar a cabo las configuraciones expuestas anteriormente.

Destacar que el objetivo principal de estas pruebas es la medición y posterior análisis del rendimiento de 4G y 5G. Para los estándares LTE-M y NB-IoT se diseñaron pruebas de conectividad, envío de mensajes y un análisis sobre la latencia y su *throughput* pero no se realizó tan en profundidad como ocurrió con las otras dos tecnologías móviles.

Las pruebas realizadas se han hecho dentro de una jaula de Faraday para no afectar al espectro radioeléctrico con nuestra propia radiación ya que se utilizan bandas de frecuencias con licencias y no se cuenta con los permisos para utilizarlas libremente. Además, se consigue que nuestra propia prueba tampoco se vea afectada por otras interferencias del exterior, consiguiendo con ello mejores resultados.

5.2.1. Pruebas para 4G y 5G

Anteriormente se han indicado las variables que se modificaran para la realización de las pruebas. En la tabla 5.5 para 4G, en la tabla 5.7 para 5G-NSA y en la tabla 5.8 para 5G. Destacar que para 5G no se usará la banda de frecuencia FR2 ya que nuestra tarjeta quectel no es compatible con estas frecuencias. En resumen para la toma de pruebas en los tres tipos de redes se cambiarán los parámetros que se muestran en la tabla 5.12.

Parámetros	Valores
Duplexación	FDD / TDD
Ancho de banda	5MHz / 10MHz 20MHz / 40MHz 50MHz
Número de antenas	SISO(1x1) / MIMO(2x2)

Tabla 5.12: Parámetros modificados en las pruebas

Se ha empleado la herramienta de iPerf3 para poder medir el rendimiento de la red. Para ello, se han establecido 10 flujos de datos simultáneos con pruebas de 15 repeticiones, posteriormente se guardan los resultados en un archivo .txt. También se utiliza la herramienta ping para medir la latencia de la red con 15 repeticiones al igual que el iPerf3. Para facilitar estas pruebas, se realizó un *script* (AnexoC) para automatizar y poder guardar los resultados en dos documentos, uno para iperf u otro para ping.

Estructuración de la información

En esta sección se explicará como se han obtenido los *dataset* para poder analizar los datos obtenidos con mayor facilidad. Para ello se analizará la información de los documentos de iPerf y de ping para estructurarla como se desee. En la Figura 5.14 se muestran un ejemplos de los documentos con los datos marcados que se van a analizar.

```
--- 192.168.3.2 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14021ms
rtt min/avg/max/mdev = 19.057/39.226/57.902/14.808 ms
PING 192.168.3.2 (192.168.3.2) 56(84) bytes of data.
64 bytes from 192.168.3.2: icmp_seq=1 ttl=64 time=28.6 ms
64 bytes from 192.168.3.2: icmp_seq=2 ttl=64 time=28.2 ms
64 bytes from 192.168.3.2: icmp_seq=3 ttl=64 time=27.8 ms
64 bytes from 192.168.3.2: icmp_seq=4 ttl=64 time=26.9 ms
64 bytes from 192.168.3.2: icmp_seq=5 ttl=64 time=25.9 ms
64 bytes from 192.168.3.2: icmp_seq=6 ttl=64 time=25.1 ms
64 bytes from 192.168.3.2: icmp_seq=7 ttl=64 time=25.0 ms
64 bytes from 192.168.3.2: icmp_seq=8 ttl=64 time=24.1 ms
64 bytes from 192.168.3.2: icmp_seq=9 ttl=64 time=23.9 ms
64 bytes from 192.168.3.2: icmp_seq=10 ttl=64 time=23.5 ms
64 bytes from 192.168.3.2: icmp_seq=11 ttl=64 time=22.9 ms
64 bytes from 192.168.3.2: icmp_seq=12 ttl=64 time=22.1 ms
64 bytes from 192.168.3.2: icmp_seq=13 ttl=64 time=20.9 ms
64 bytes from 192.168.3.2: icmp_seq=14 ttl=64 time=19.0 ms
64 bytes from 192.168.3.2: icmp_seq=15 ttl=64 time=18.0 ms
```

[ID]	Interval		Transfer	Bitrate	Retr	Cwnd
[5]	0.00-1.00	sec	3.26 MBytes	27.4 Mbits/sec	0	198 KBytes
[7]	0.00-1.00	sec	4.46 MBytes	37.4 Mbits/sec	0	287 KBytes
[9]	0.00-1.00	sec	4.40 MBytes	36.9 Mbits/sec	0	286 KBytes
[11]	0.00-1.00	sec	3.17 MBytes	26.6 Mbits/sec	0	178 KBytes
[13]	0.00-1.00	sec	3.05 MBytes	25.6 Mbits/sec	0	223 KBytes
[15]	0.00-1.00	sec	3.20 MBytes	26.9 Mbits/sec	0	223 KBytes
[17]	0.00-1.00	sec	2.49 MBytes	20.9 Mbits/sec	0	156 KBytes
[19]	0.00-1.00	sec	2.61 MBytes	21.9 Mbits/sec	0	167 KBytes
[21]	0.00-1.00	sec	2.68 MBytes	22.5 Mbits/sec	0	175 KBytes
[23]	0.00-1.00	sec	4.43 MBytes	37.2 Mbits/sec	0	337 KBytes
[SUM]	0.00-1.00	sec	33.8 MBytes	283 Mbits/sec	0	

Figura 5.14: Ejemplo de datos de documentos iPerf y ping

En el documento iPerf se utilizará la última fila de cada iPerf realizado, ya que se trata de la suma de *throughput* enviado a través de los 10 flujos. Además del intervalo de cuando se realizó y la cantidad de datos enviados (Transfer). Con respecto al ping se utilizará la primera fila, donde nos indica el tiempo de ida y vuelta (RTT) mínimo, máximo, su media y su desviación.

Se han utilizado dos códigos de python para analizar y organizar la información de estos documentos y poder obtener los datos de una manera más sencilla y estructurada. En el Anexo D se pueden ver ambos códigos *parseo_iperf.py* y *parseo_ping.py* que emplean la libreria de Python Pandas ya que ofrece una amplia gama de herramientas para estudiar y manipular los datos. La solución obtenida de estos códigos se guardará en dos documentos .csv. Donde se tiene un primera fila con el tipo de datos de la columna. Un ejemplo del resultado obtenido se puede observar en la Figura 5.15.

```
Tecnologia,Duplex,BW,MIMO,Interv,Transf,Throughput,PcktRetr  Tecnologia,Duplex,BW,MIMO,Ptx,Prx,Pl,Time,RTTmin,RTTavg,RTTmax,RTTmdev
4G,FDD,10,MIMO,0.00-1.00,18.8,158,0                         4G,FDD,10,MIMO,15,15,0,14022,20.847,29.746,38.829,4.992
4G,FDD,10,MIMO,1.00-2.00,9.07,76.1,0                        4G,FDD,10,MIMO,15,15,0,14017,20.898,30.259,39.018,6.087
4G,FDD,10,MIMO,2.00-3.00,11.2,93.8,0                        4G,FDD,10,MIMO,15,15,0,14019,19.978,29.169,38.874,5.721
4G,FDD,10,MIMO,3.00-4.00,9.57,80.3,0                        4G,FDD,10,MIMO,15,15,0,14020,20.025,29.528,38.820,5.545
4G,FDD,10,MIMO,4.00-5.00,9.13,76.6,0                        4G,FDD,10,MIMO,15,15,0,14022,19.832,28.407,37.868,5.271
4G,FDD,10,MIMO,5.00-6.00,8.64,72.5,0                        4G,FDD,10,MIMO,15,15,0,14019,20.024,29.281,38.896,5.729
4G,FDD,10,MIMO,6.00-7.00,10.1,84.5,0                        4G,FDD,10,MIMO,15,15,0,14019,20.878,30.368,39.370,5.561
4G,FDD,10,MIMO,7.00-8.00,9.45,79.2,0                        4G,FDD,10,MIMO,15,15,0,14023,20.895,29.765,38.973,5.267
4G,FDD,10,MIMO,8.00-9.00,9.88,82.9,0                        4G,FDD,10,MIMO,15,15,0,14022,20.949,29.799,38.915,5.041
4G,FDD,10,MIMO,9.00-10.00,9.51,79.8,0                       4G,FDD,10,MIMO,15,15,0,14021,19.462,29.086,38.212,5.448
4G,FDD,10,MIMO,0.00-1.00,10.6,89.1,0                        4G,FDD,10,MIMO,15,15,0,14024,19.947,28.822,37.950,4.934
4G,FDD,10,MIMO,1.00-2.00,10.4,87.1,0                        4G,FDD,10,MIMO,15,15,0,14018,20.061,29.350,38.939,5.872
4G,FDD,10,MIMO,2.00-3.00,9.63,80.8,0                        4G,FDD,10,MIMO,15,15,0,14015,20.010,27.707,38.949,5.318
4G,FDD,10,MIMO,3.00-4.00,10.3,86.5,0                        4G,FDD,10,MIMO,15,15,0,14021,20.828,29.964,39.021,5.444
4G,FDD,10,MIMO,4.00-5.00,9.94,83.4,0                        4G,FDD,10,MIMO,15,15,0,14018,19.983,29.413,39.036,5.968
4G,FDD,10,MIMO,5.00-6.00,8.95,75.1,0                        4G,FDD,10,SISO,15,15,0,14019,20.907,30.517,38.983,5.450
4G,FDD,10,MIMO,6.00-7.00,10.0,83.9,0                        4G,FDD,10,SISO,15,15,0,14018,19.460,28.526,37.960,5.736
4G,FDD,10,MIMO,7.00-8.00,9.20,77.1,0                        4G,FDD,10,SISO,15,15,0,14017,20.915,29.913,39.091,5.624
4G,FDD,10,MIMO,8.00-9.00,9.82,82.4,0                        4G,FDD,10,SISO,15,15,0,14016,20.071,28.544,39.038,6.199
4G,FDD,10,MIMO,9.00-10.00,8.58,71.9,0                       4G,FDD,10,SISO,15,15,0,14022,20.803,29.823,38.978,5.468
```

Figura 5.15: Ejemplo de datos parseados

5.2.2. Pruebas para LTE-M y Nb-IoT

En la realización de las pruebas de LTE-M y NB-IoT se llevo a cabo una prueba de conexión entre la estación base y la mota pycom. En primer lugar simplemente se realizó la conexión de una de las tecnologías y en otra prueba con configuración distinta la otra tecnología.

Al comprobar que estas pruebas eran exitosas, se planteó el envío de mensajes TCP a través de un socket. Para ello, en nuestro código de Python se debe indicar la dirección IP del servidor y el puerto al que se debe conectar. En el servidor se debe indicar también el puerto de conexión y para agilizar las pruebas se realizó un *script*, para lanzar el servidor con el socket indicando el puerto que queramos sin tener que entrar en su archivo de configuración. En la tabla 5.13 se muestran las direcciones IP asignadas a los clientes y servidor de la conexión socket. Destacar que se ha utilizado el puerto 6000 para LTE-M y el 6001 para NB-IoT. Además, se ha empleado

la banda de frecuencia 20, compatible con nuestras motas.

Elemento	Dirección IP
Estación base	192.168.3.1
Mota Pycom LTE-M	192.168.3.2
Mota Pycom Nb-IoT	192.168.3.10

Tabla 5.13: Direcciones IP, escenario IoT

Por concluir, la última prueba realizada fue la de probar la conexión de ambas tecnologías y el envío de mensajes a la vez hacia el servidor a través de los puertos indicados. Para ello se utilizó un archivo de configuración de la estación base que lanzara los servicios de LTE-M y NB-IoT. Uno de los códigos de configuración empleados para la mota es el siguiente:

```
from network import LTE
import time
import socket

    # Inicia el modem
lte = LTE(debug=True)


    # Conexion con la red, utilizando banda 20
lte.attach(band=20, apn="internet")
print("attaching...",end='')

while not lte.isattached(): # Comprueba la conexion fisica
    time.sleep(0.25)

    print('.',end='')
    print(lte.send_at_cmd('AT!="fsm"'))         # Obtener informacion
        sobre los estados y transiciones de FSM
print("Attached!")

# Conectar con red LTE
lte.connect()
print("connecting [##",end='')

while not lte.isconnected():
    time.sleep(0.25)
    print('#',end='')

    print(lte.send_at_cmd('AT!="fsm"'))
    ip_address = "192.168.3.1"
    print(lte.send_at_cmd('AT+PING="192.168.3.1",1,32'))

print("] connected!")
print("Conexion a la red LTE establecida.")



# Conexion al socket
while lte.isconnected():

```

```
40      s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
41      # Obtencion IP y puerto
42      ip_address = "192.168.3.1" #Indica la direccion IP del server
43      port = 6001
44      print('Esperando a conectartse')
45      s.connect((ip_address, port))
46      print('socket conectado')
47
48      # Enviar datos al servidor
49      while True:
50          message = 'Hola desde la mota NB-IoT'
51          s.sendall(message.encode())
52
53          # Recibir respuesta del servidor
54          response = s.recv(1024)
55          print('Respuesta del servidor:', response.decode())
56
57      # Cerrar la conexion
58  s.close()
```

Capítulo 6

Análisis de resultados

En esta sección se analizarán todos los resultados de las diferentes pruebas realizadas para las tecnologías móviles de 4G y 5G. Se utilizará los documentos con los datos ya estructurados, facilitando el análisis de rendimiento mediante las gráficas de CDF, obtenidas a partir de un código de Python (Anexo E). Para LTE-M y NB-IoT se analizará la conexión de ambas tecnologías al mismo tiempo, así como los mensajes enviados para esta, junto a un pequeño vistazo a su *throughput* y su latencia.

6.1. Análisis 4G y 5G

Se realizará un análisis individual de cada tecnología comparando el *throughput* en megabits por segundo (Mbits/s) y de la latencia en milisegundos (ms).

6.1.1. Análisis del Throughput

A continuación, se examinará cada tecnología por separados para cada uno de los parámetros que se han cambiado, empezando por el número de antenas utilizadas en la Figura 6.1. Se puede comprobar que en las tres tecnologías al utilizar un número mayor de antenas (MIMO) se obtiene un mayor *throughput* al poder recibir y transmitir múltiples flujos de forma simultanea que si simplemente se utiliza una antena para transmitir y otra para recibir datos.

Para 4G como para 5G-NSA si se utiliza la duplexación de TDD se obtiene un rendimiento más bajo que si se utiliza FDD, ya que este último utiliza bandas de recepción y transmisión separadas además de una mejor optimización de capacidad y flexibilidad consiguiendo menos interferencias,

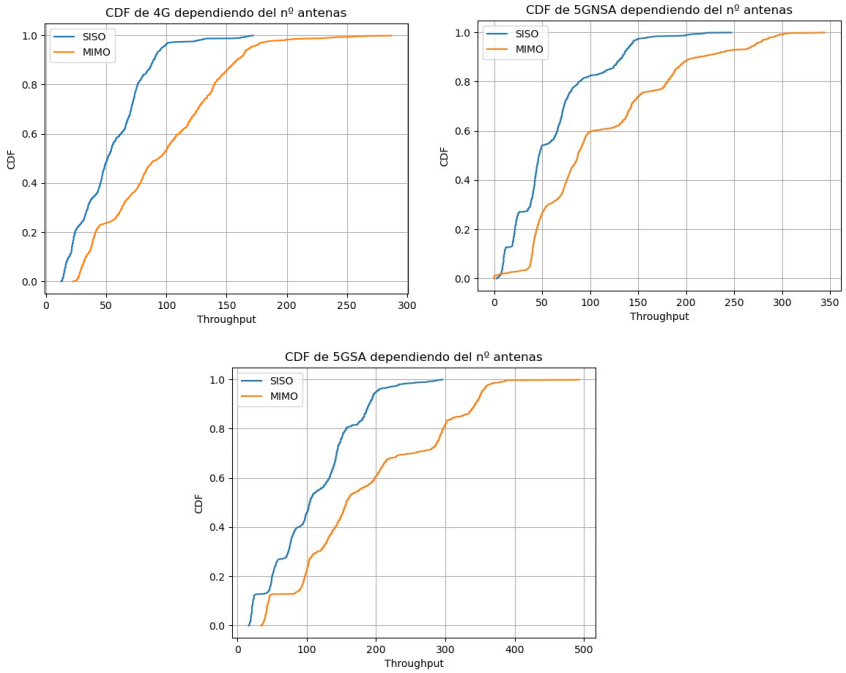

Figura 6.1: CDF de Throughput dependiendo del nº antenas

por consiguiente un *throughput* mayor. Pero para 5G-SA no ocurre este caso sino que se obtiene un mejor rendimiento con TDD, se debe a que se ha utilizado la banda 78 con un *subcarrier spacing* de 30 kHz consiguiendo que haya un menor espacio entre las subportadoras y con ello una mayor capacidad y mejor rendimiento que con FDD que utilizó 15 kHz, ya que nuestra tarjeta Quectel no soporta los 30 KHz para esta duplexación. Esta deducción se puede obtener a partir de la gráfica mostrada en la Figura 6.2.

Por último se compara el *throughput* dependiendo del ancho de banda. Con la tecnología móvil 4G se realizaron mediciones utilizando diferentes anchos de bandas, incluyendo 5 MHz, 10 MHz, 15 MHz y 20 MHz. Para 5G-NSA se emplearon las mismas capacidades excepto cuando se utilizó la duplexación de TDD que no se usó el ancho de banda de 5 MHz ya que nuestra tarjeta quectel no soporta esta capacidad con la banda 41. Al igual ocurre con 5G-SA pero en este caso se ha empleado la banda 78 pero nuestra quectel no soporta un ancho de banda menor de 20 MHz, por tanto, se utilizaron los anchos de banda de 20 MHz, 40 MHz y 50 MHz. Al utilizar un ancho de banda mayor, se puede transmitir una mayor cantidad de datos de manera simultanea lo que significa un *throughput* superior, como se cumple en la Figura 6.3 para las tres tecnologías.

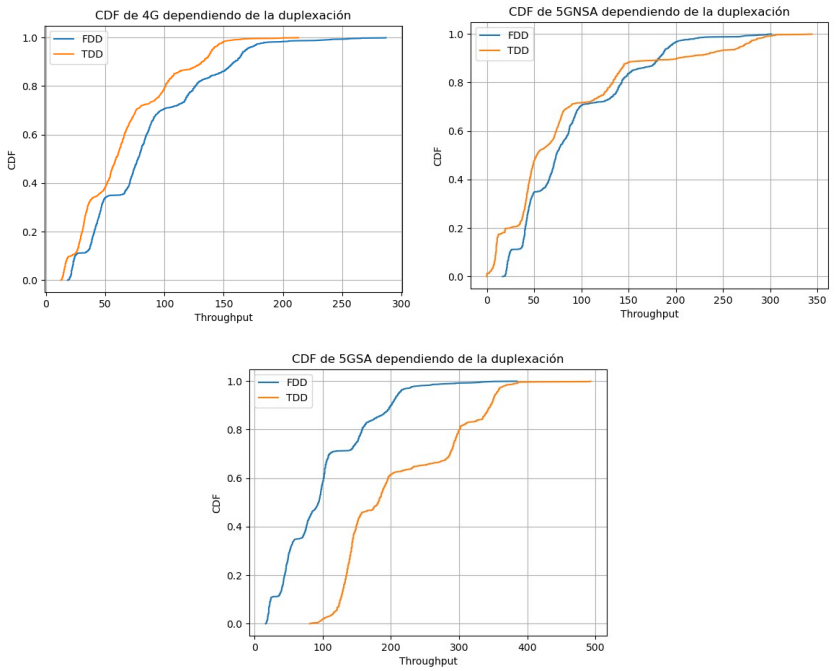

Figura 6.2: CDF de Throughput dependiendo de la duplexación

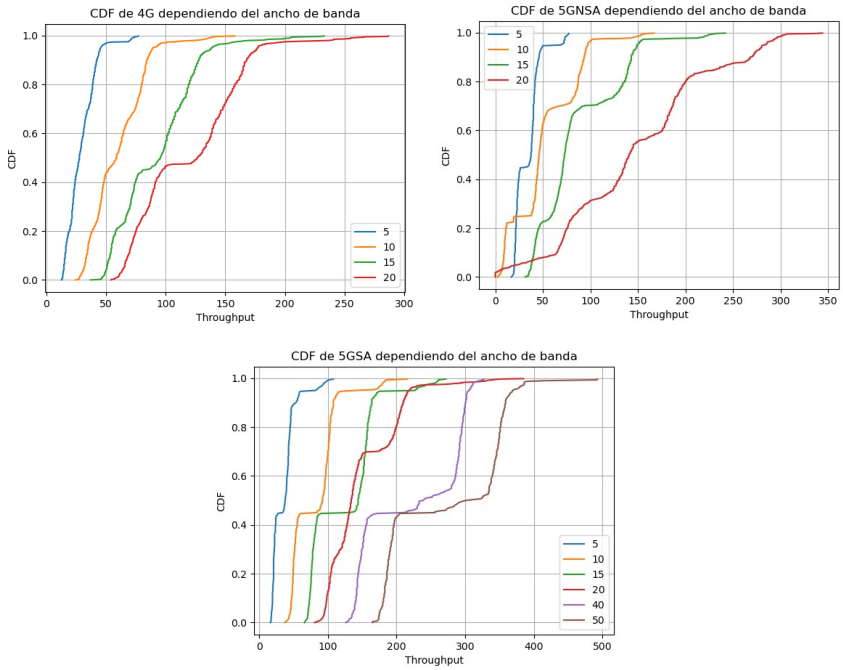

Figura 6.3: CDF de Throughput dependiendo del ancho de banda

6.1.2. Análisis de latencia

Al realizar ping se obtiene el tiempo de ida y vuelta (RTT). Por tanto, este número se divide entre dos para obtener la latencia (únicamente el tiempo de ida). Dependiendo del número de antenas la diferencia de latencia es mínima, aumentando ligeramente cuando se utiliza MIMO, como se pude observar en la Figura 6.4.

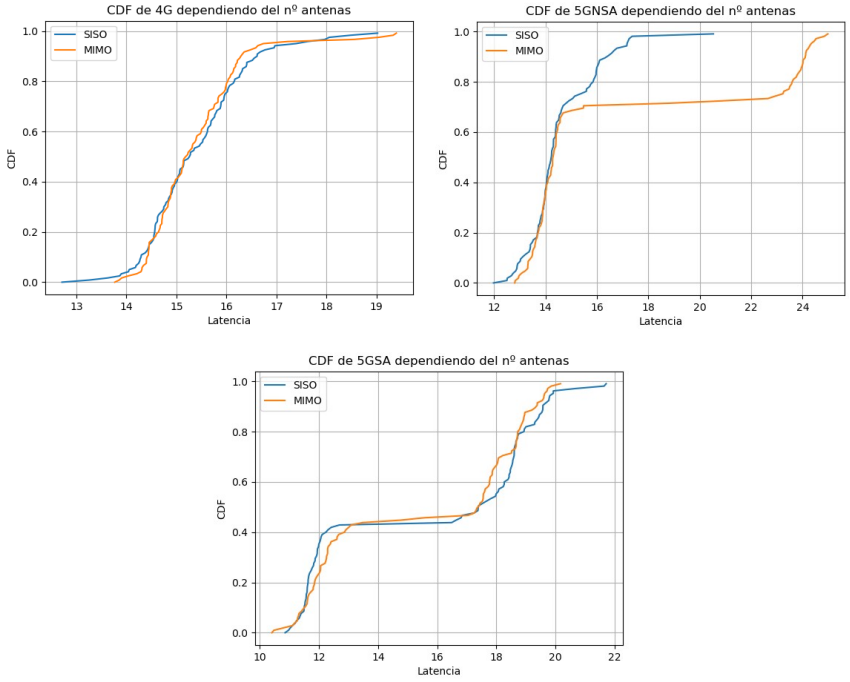

Figura 6.4: CDF de latencia dependiendo del nº antenas

Al comparar la duplexación tanto para 4G y 5G-NSA se obtiene una menor latencia para FDD, en la Figura 6.5 se indica esta afirmación. Con TDD se va alternando en el tiempo la transmisión y recepción lo que añade una pequeña demora en el tiempo, en cambio con FDD se transmite y recibe información de forma simultanea al asignar frecuencias distintas para cada acción. En cambio para 5G-SA con FDD tiene una mayor latencia por el uso de un *subcarrier spacing* de 15 kHz en vez de 30 kHz, afectando considerablemente al tiempo.

Por último se compara el retardo de la red entre las diferentes tecnologías dependiendo del ancho de banda utilizado. En 4G la latencia no se ve aumentada con el ancho de banda ya que al aumentar la cantidad de datos que se puede enviar no debe interferir con la latencia. En la Figura 6.6

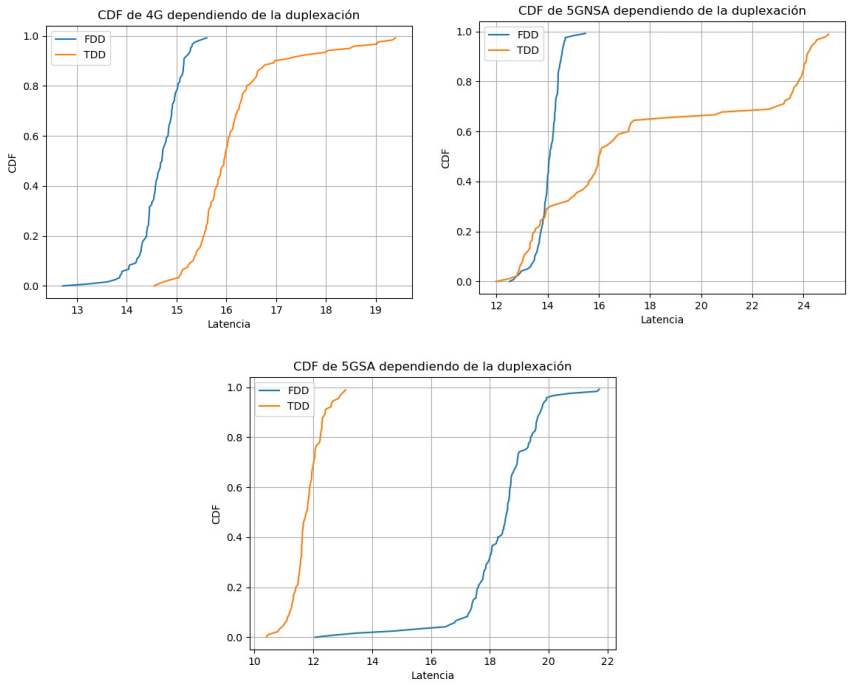

Figura 6.5: CDF de latencia dependiendo de la duplexación

se puede ver como prácticamente no hay cambio entre los distintos ancho de bandas, se puede ver un pequeño aumento de latencia en la tecnología 5G-NSA ya que esos anchos de bandas dan problemas con nuestra tarjeta quectel. Para 5G-SA se muestra que para los ancho de banda utilizados en FDD (5,10,15,20 MHz) se tiene un tiempo mucho mayor. Como se explico anteriormente, se concluye que la red de 5G no se encuentra totalmente optimizada para FDD debido a la utilización de un *subcarrier spacing* de 15 kHz.

Tal como se mencionó anteriormente se ha conseguido obtener el RTT máximo junto a la media y el tiempo mínimo. En la Figura 6.7 se emplean estos valores para realizar una comparativa entre las distintas tecnologías. Se demuestra que la latencia mínima es bastante estable en 4G y 5G-SA, consiguiendo prácticamente para todas las pruebas una media de mínimo por debajo de 10 ms.

6.2. Análisis entre las tecnologías 4G y 5G

En esta sección se comparará las tecnolgías móviles de 4G y 5G en una misma gráfica para poder obtener conclusiones al compararlas entre ellas.

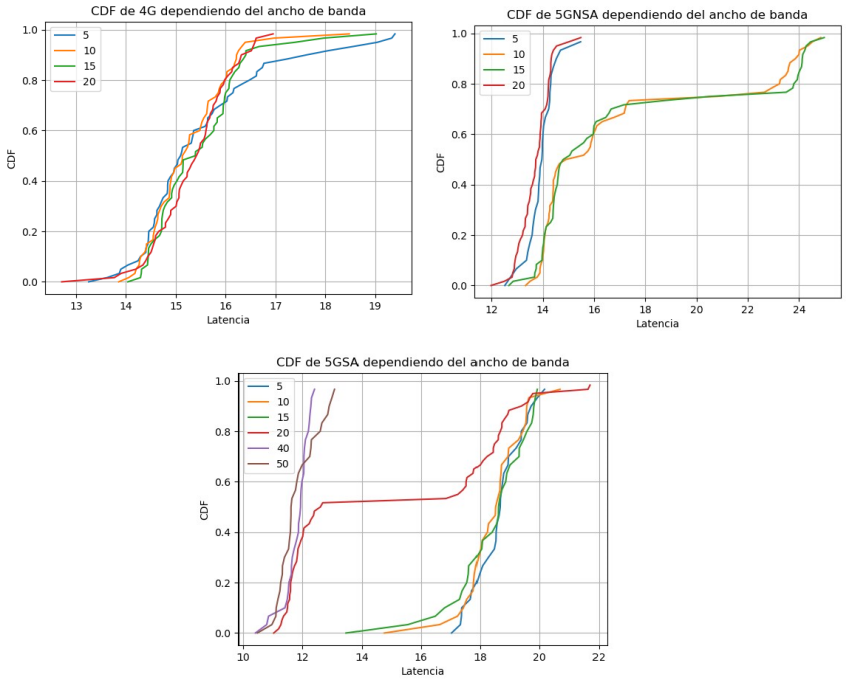

Figura 6.6: CDF de latencia dependiendo del ancho de banda

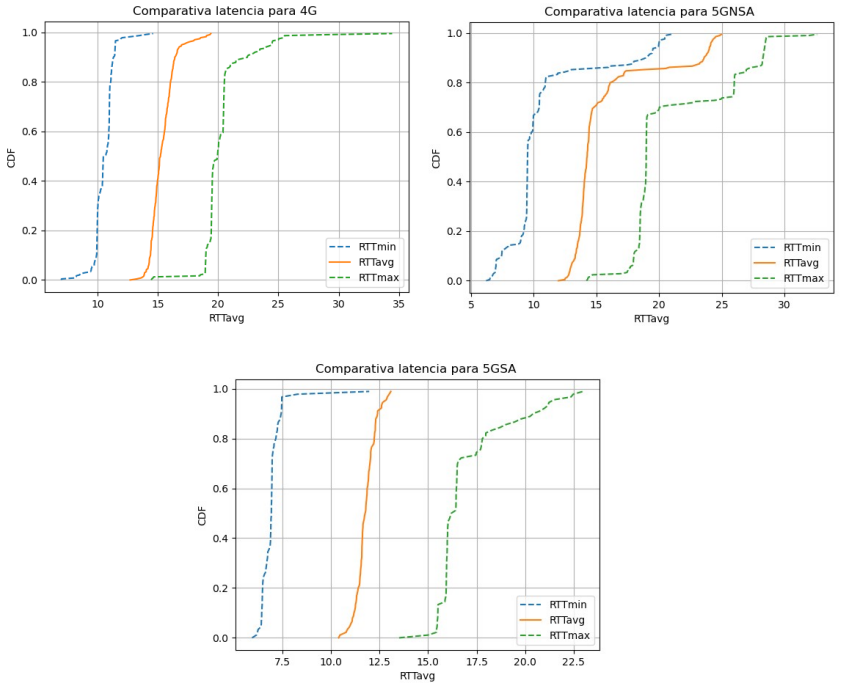

Figura 6.7: CDF de latencia mínima, media y máxima

6.2.1. Análisis del throughput

En primer lugar se analizará los datos de iPerf, en concreto el *throughput* medio de todas las tecnologías y así conseguir una idea general sobre que tecnología tiene un rendimiento mayor. En la Figura 6.8 se puede ver claramente que al emplear la tecnología 5G-SA se obtiene un *throughput* mayor que con cualquiera de las otras dos tecnologías.

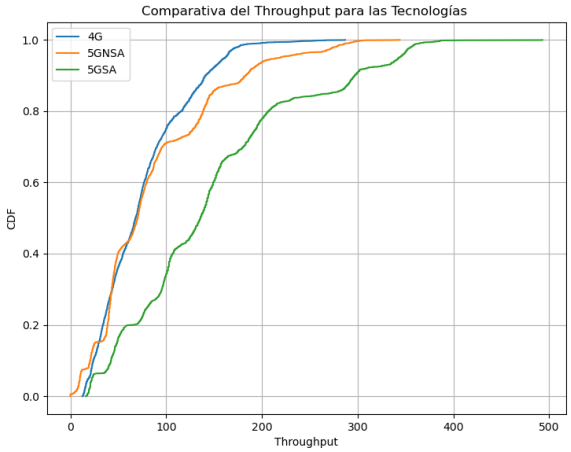

Figura 6.8: CDF de Throughput medio

Si se comparan todas la tecnologías según el número de antenas, se puede ver claramente en la Figura 6.9 que con la tecnología de 5G-SA utilizando MIMO se obtiene el *throughput* más elevado.

Para el ancho de banda todos tiene un resultado bastante similar como se muestra en la Figura 6.10 excepto para 5G-SA con TDD que consigue una media de 360 Mbits/s, sobresaliendo de las demás medidas.

La gráfica del *throughput* dependiendo del ancho de banda mostrada en la Figura 6.11 puede llegar a ser algo caótica por su gran cantidad de datos, pero se puede ver claramente para todas las tecnologías como a un mayor ancho de banda se obtiene un mayor rendimiento, consiguiendo despuntar como ya se va comentando previamente 5G-SA.

6.2.2. Análisis de la latencia

Al igual que con el iPerf, en este caso se analizará la latencia de la red. En la Figura 6.12 se puede ver la latencia media dependiendo de cada tec-

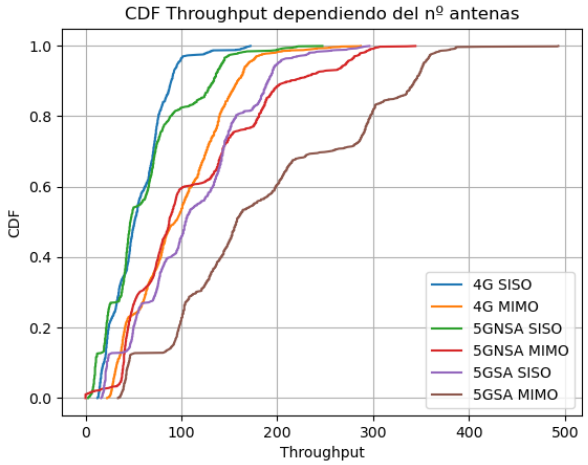

Figura 6.9: CDF de Throughput dependiendo del nº antenas

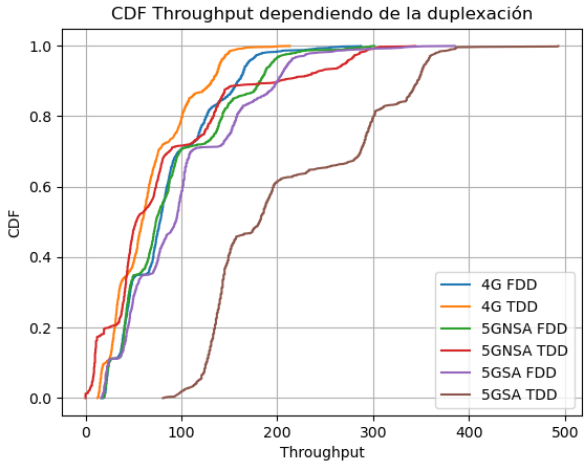

Figura 6.10: CDF de Throughput dependiendo de la duplexación

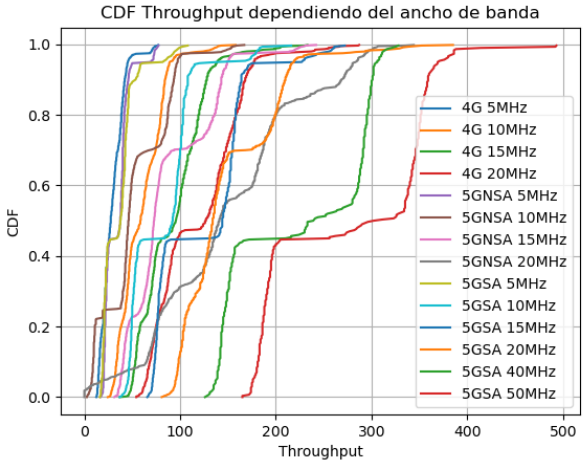

Figura 6.11: CDF de Throughput dependiendo del ancho de banda

nología, donde 5G-SA obtiene mejores resultados al conseguir una menor latencia que las otras dos tecnologías, pero no es nada estable lo que se deduce que cuenta con resultados con una latencia mucho mayor. En cambio 4G cuenta con una latencia un poco más superior pero siendo mucho más estable a lo largo de la realización de las pruebas. Esto demuestra que las redes 5G-SA pueden mejorar lo existente, pero actualmente no se encuentran lo suficientemente desarrolladas.

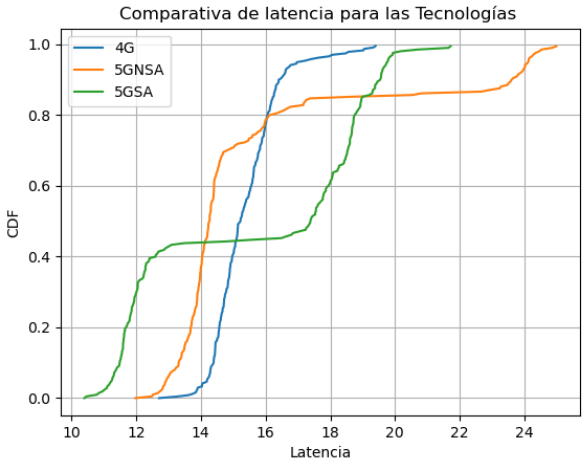

Figura 6.12: CDF de latencia media

Al cambiar el número de antenas de SISO una antena para transmitir y

una antena para recibir a MIMO dos para transmitir y dos para recibir se ve que realmente no hay una variación entre ambos, sino que la verdadera diferencia se debe a la tecnología empleada. Con 5G-SA se consigue una menor latencia pero no es nada estable debido a los altos valores de FDD explicados anteriormente. En la Figura 6.13 se muestra claramente esta afirmación.

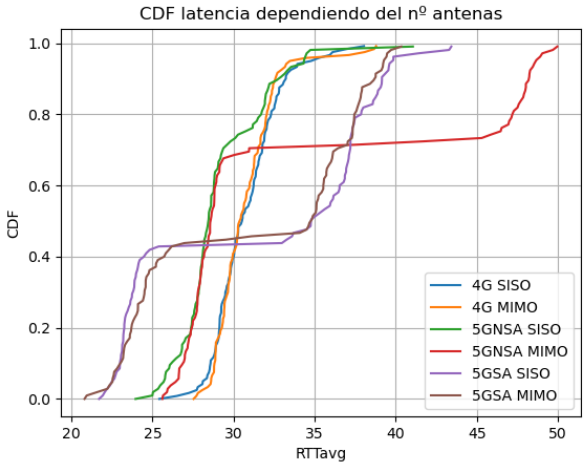

Figura 6.13: CDF de latencia dependiendo del nº antenas

Cuando se analiza la latencia dependiendo de la duplexación se debe tener en cuenta que las pruebas realizadas para 5G-SA con FDD no se encuentran totalmente optimizadas para su latencia. Igualmente, la tecnología móvil que obtiene una menor latencia es esta misma pero empleando la duplexación de TDD. Le sigue 5G-NSA con FDD y 4G FDD en orden ascendente respectivamente como se observa en la Figura 6.14.

Tras dividir los resultados obtenidos dependiendo de la tecnología y el ancho de banda se consigue la CDF de la Figura 6.15. Se puede ver claramente que la tecnología de 5G-SA para estos parámetros sigue siendo la que tiene una menor latencia. Sobre todo las de mayor ancho de banda, pero como se indicó previamente, el ancho de banda no interfiere con la latencia. Y no todas las medidas de una misma tecnología tienen una misma latencia debido a la optimización de la red o las compatibilidades con la quectel.

6.3. Análisis LTE-M y NB-IoT

A diferencia de las tecnologías anteriores para LTE-M como NB-IoT se utilizó las motas pycom como nuestro UE. Se realizaron pruebas de conec-

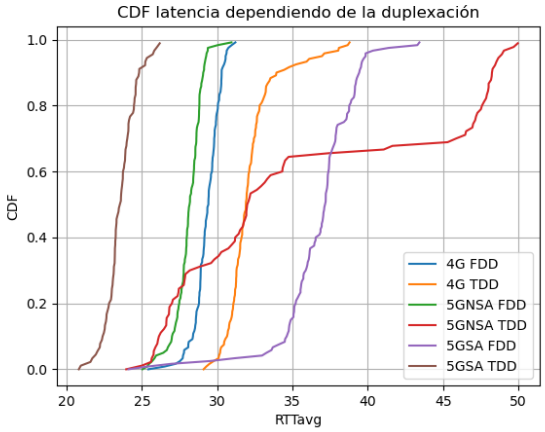

Figura 6.14: CDF de latencia dependiendo de la duplexación

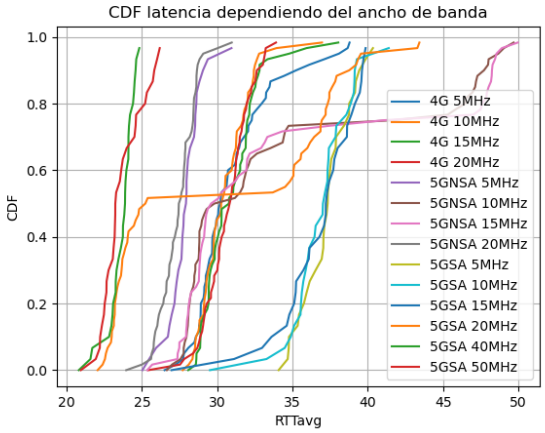

Figura 6.15: CDF de latencia dependiendo del ancho de banda

tividad enviando mensajes TCP mediante un socket desde la estación base hacia la mota pycom, servidor y cliente respectivamente.

Gracias a la interfaz web de Amarisoft se ha capturado la conexión NB-IoT, en la Figura 6.16 se observa una parte de los mensajes enviados y se ve como muchos de ellos están duplicados, debido a intentos de conexión del sistema.

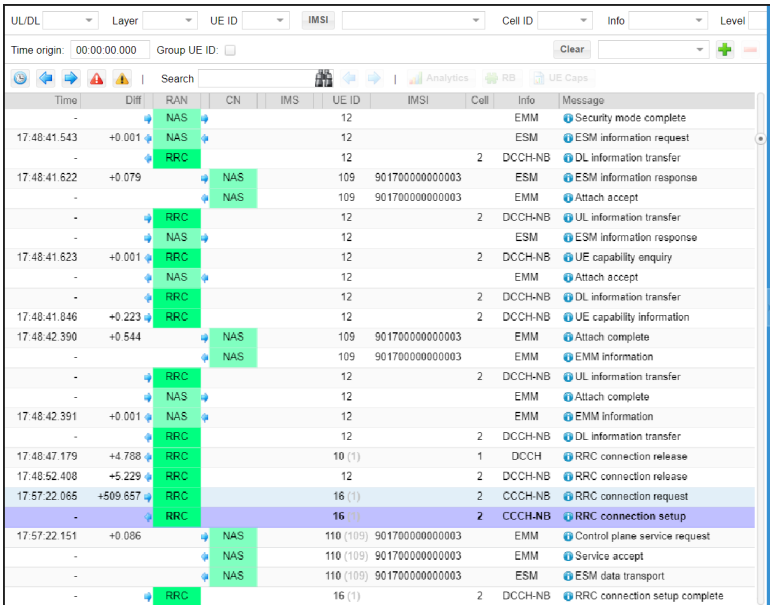

Figura 6.16: Mensajes de conexión Nb-Iot

En la Figura 6.17 se muestra un diagrama de secuencia eliminando los mensajes repetidos para poder visualizar de una mejor forma la conexión completa entre la estación base y la mota pycom. Cabe destacar que la conexión se realiza igual para LTE-M y NB-IoT son iguales.

A continuación se analizará cada uno de los mensajes con una pequeña explicación:

- **RRC connection request:** Mensaje desde el UE hacia la estación base para solicitar una conexión.
- **RRC connection setup:** Mensaje desde la estación base en respuesta al *RRC connection request* indicando el establecimiento de la conexión RRC.
- **Attach Request:** Mensaje para solicitar el registro en la red.

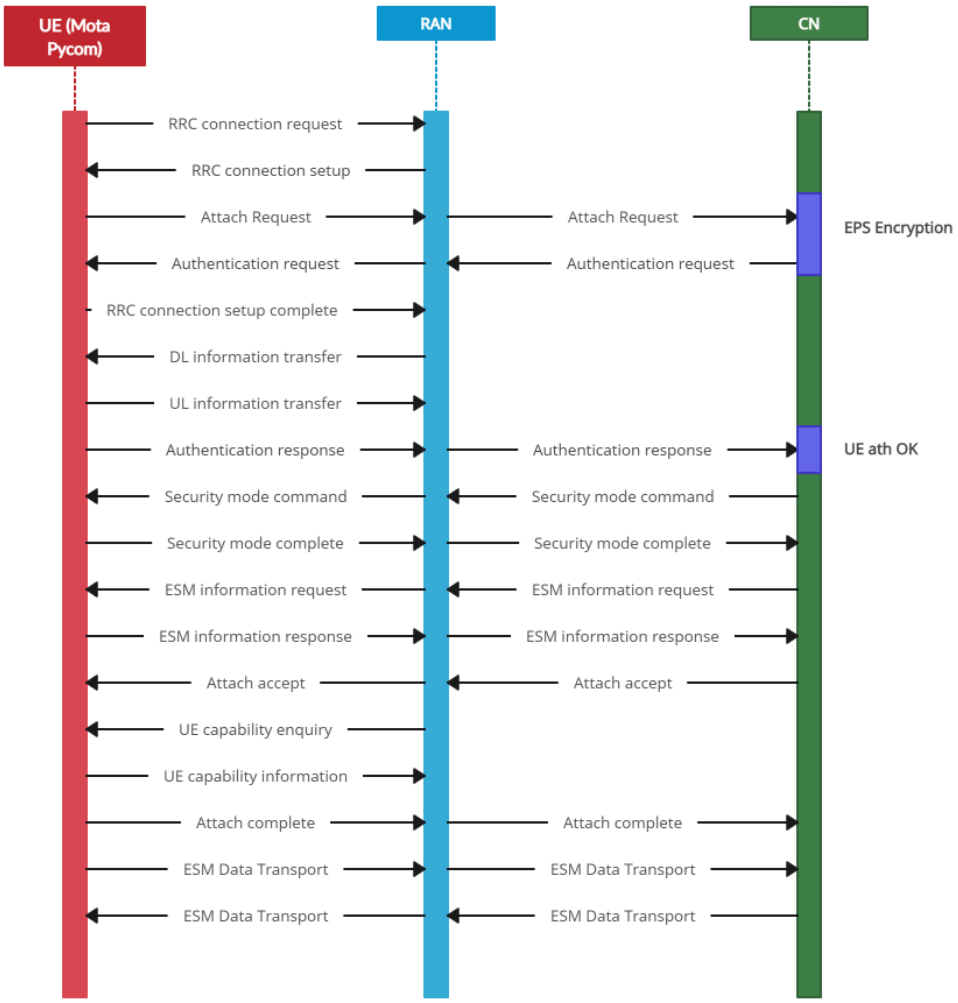

Figura 6.17: Mensajes resumidos de conexión NB-Iot

- **Authetication Request:** Se solicita la autenticación del dispositivo para poder aceptar el registro en la red.
- **RRC connection setup complete:** Mensaje para la estación base indicando que la conexión RRC ha sido exitosa.
- **DL information transfer:** Mensaje de envío de información desde la estación base al UE (*downlink*).
- **UL information transfer:** Mensaje de envío de información desde el UE a la estación base (*uplink*).
- **Authetication Response:** Envía la información de autenticación solicitada en el *Authetication Request*.
- **Security mode command:** Mensaje enviado hacia el UE para comprobar que modo de seguridad se va a establecer en la conexión.
- **Security mode complete:** Mensaje que indica que se ha establecido un modo de seguridad.
- **ESM information request:** La red lo utiliza para recuperar información ESM, como *Access Point Name* (APN) e información de configuración de protocolos si el UE indica que cuenta con información que debe cifrarse.
- **ESM information response:** Mensaje enviado a la red que incluye la opciones de configuración de protocolo que deben ser protegidas al transferirse.
- **Attach Accept:** Se confirma el registro de la red.
- **UE capability enquiry:** La red pide las capacidades de la mota Pycom.
- **UE capability information:** El UE responde con la información de las capacidades, como se mostró previamente en la Figura 5.13.
- **Attach complete:** Mensaje del UE indicando que se ha completado el registro y es correcto.
- **ESM Data transport:** Mensaje que contiene los datos enviados, en nuestro caso mensajes TCP.

A través de Tshark, pudimos analizar el tráfico y obtener los mensajes enviados durante la conexión que se han podido analizar mediante Wireshark. En la Figura 6.18 se muestra un diagrama donde se ve el envío de mensajes TCP entre NB-IoT con la IP 192.168.3.10 hacía la estación base

(192.168.3.1) y esta la mota LTE-M (192.168.3.2), consiguiendo con una sola estación base poder desplegar dos tecnologías móviles de baja latencia a la vez.

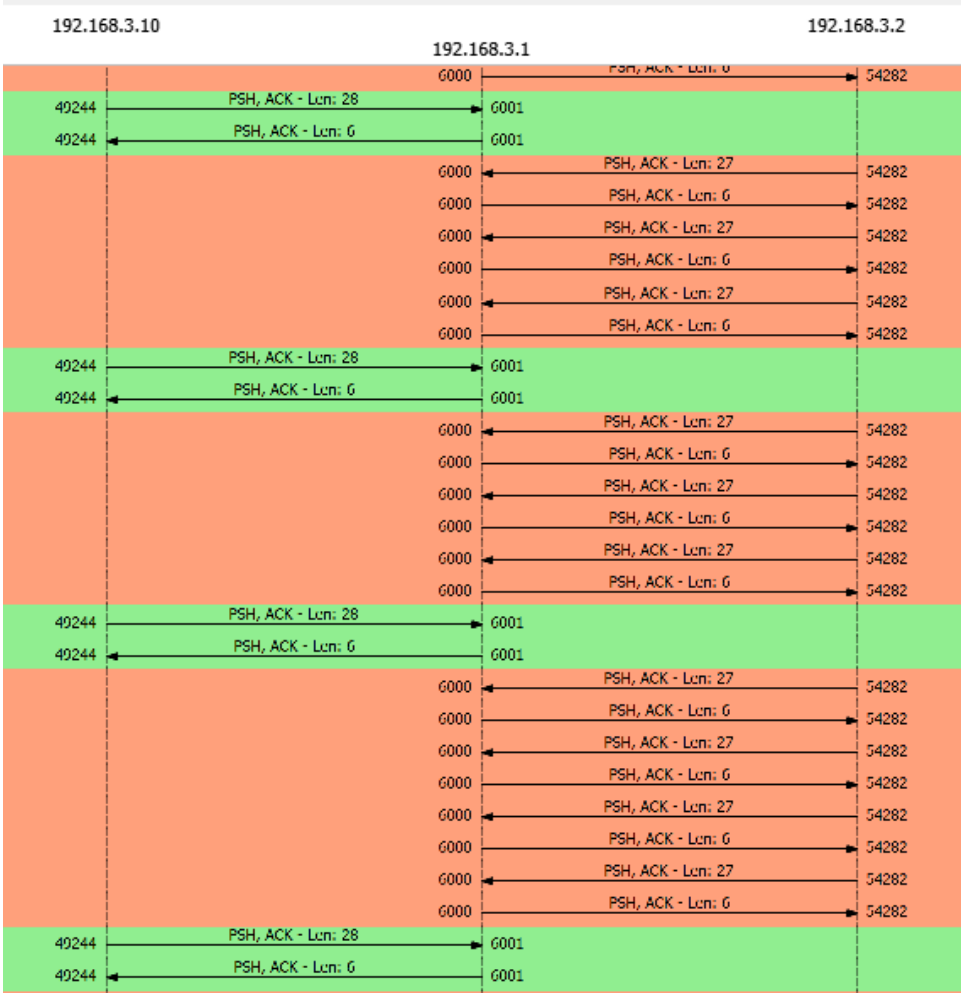

Figura 6.18: Mensajes TCP de LTE-M y Nb-IoT

En la Figura 6.19 se observa los Bytes/seg enviados a lo largo de nuestra conexión. Se puede observar como LTE-M manda mensajes únicamente durante un tiempo, en su código de Python, se le asignó que enviase solo 100 mensajes, en cambio en NB-IoT se configuró para enviar mensajes de forma indefinida. LTE-M como indica la teoría tiene una tasa de transferencia mayor que NB-IoT. Llegando a 12.5 Kbits/seg y 5.5Kbits/seg respectivamente, una tasa a priori bastante baja pero se debe tener en cuenta que nuestro principal objetivo era la conectividad de ambas tecnologías a la vez y no la optimización de la velocidad de envío de datos.

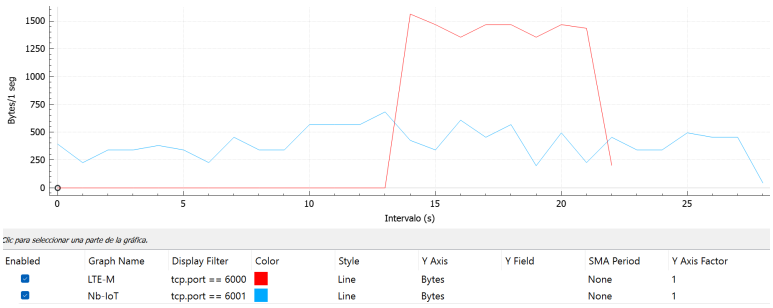

Figura 6.19: Bytes enviados durante la conexión

En las Figuras 6.20 y 6.21 se muestra el rendimiento medio en bits/seg, donde destaca que LTE-M es mucho más estable rondando los 600 bits/seg, en cambio NB-IoT tiene un *throughput* mucho más inestable, con máximos de 300 bits/seg. Con la latencia ocurre algo similar, en las Figuras 6.22 y 6.23 se ve como LTE-M tiene una latencia de unos 80 ms la mayor parte del tiempo pero NB-IoT como con su rendimiento tiene una latencia muy poco estable, llegando incluso a 700 ms. Se puede concluir, que NB-IoT al ser una tecnología utilizada para enviar paquetes pequeños en un tiempo concreto, no es su mayor prioridad la latencia o el *throughput*. Igualmente ocurre con LTE-M pero en este caso es más estable ya que está pensando para enviar un mayor número de datos.

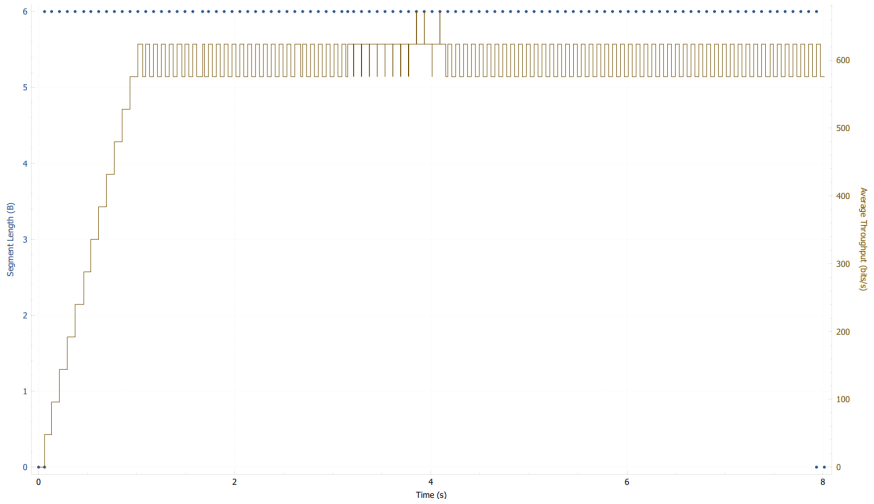

Figura 6.20: Rendimiento medio de LTE-M

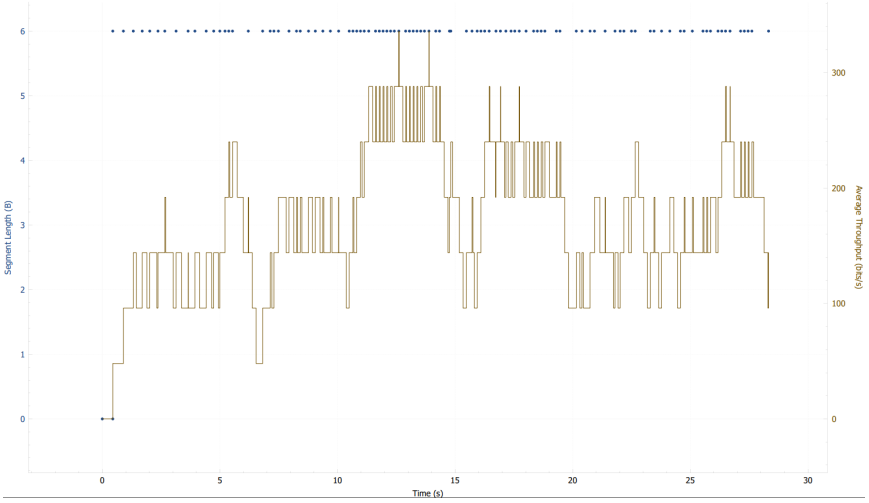

Figura 6.21: Rendimiento medio de NB-IoT

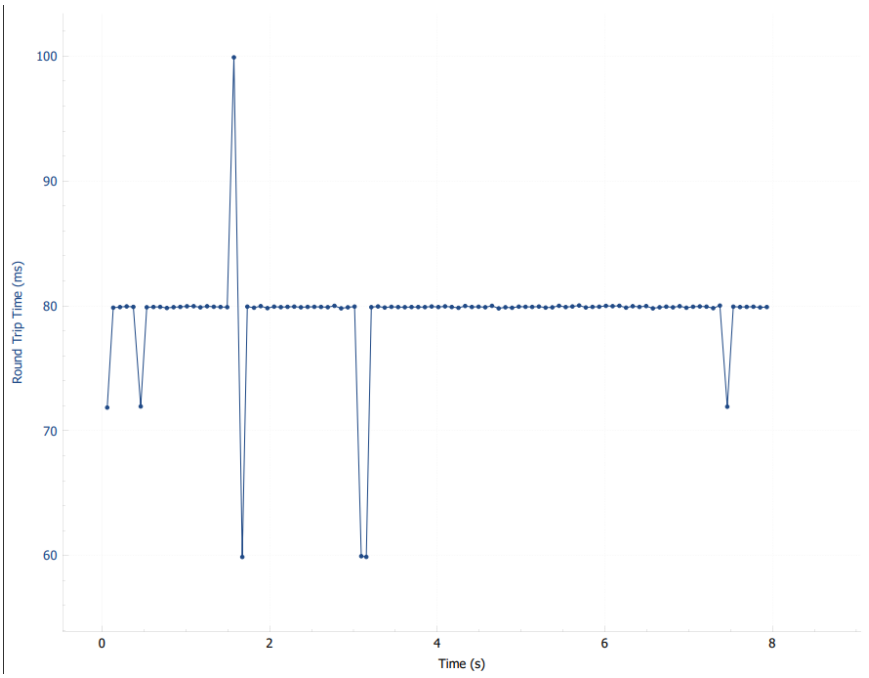

Figura 6.22: Latencia de LTE-M

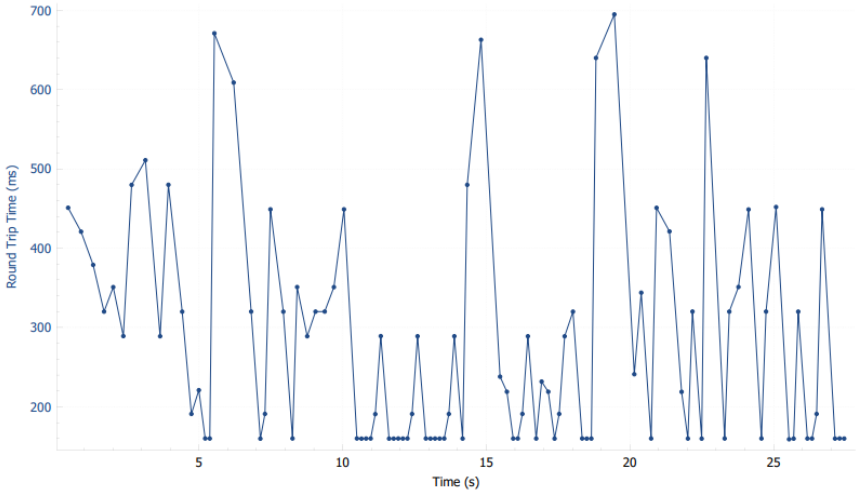

Figura 6.23: Latencia de NB-IoT

Capítulo 7

Conclusiones y líneas futuras

En esta sección se exponen los objetivos cumplidos en el proyecto así como las conclusiones obtenidas tras analizar los resultados obtenidos. También se indicarán los trabajos futuros para mejorar o complementar este proyecto.

7.1. Conclusiones

El principal objetivo de este proyecto era la de configurar una red y poder realizar pruebas de conectividad para LTE, 5G-NSA, 5G-SA, LTE-M y NB-IoT junto a la obtención y análisis de resultados para cada una de estas tecnologías móviles. Consiguiendo con gran éxito todos los objetivo propuestos en la sección 1.2. Al final se nombran los trabajos futuros de mejora que se pueden realizar a partir de este proyecto.

Para ello, se estudiaron las principales alternativas para desplegar estas redes, donde se concluyó que la mejor opción era el software Amarisoft debido a todas las funcionalidades que nos presta. Consiguiendo con ello poder desplegar todas las tecnologías (4G, 5G-NSA, 5G-SA, LTE-M y NB-IoT) tras examinar los archivos de configuración y entender cual es la mejor forma de desarrollar el proyecto. De igual forma, se tuvo que configurar y estudiar los dispositivos de conexión (UE) ya que nos dan limitaciones como las bandas y capacidades que soportan.

En la realización de las pruebas se estudiaron los parámetros de configuración que eran de interés para la obtención de resultados sobre el *throughput* y latencia. Se utilizó las herramienta de iPerf para enviar un flujo de datos TCP para poder medir el rendimiento de la red. La herramienta de Ping se empleó conseguir el tiempo de ida y vuelta (RTT) de los paquetes, además de la obtención de otros parámetros de interés descritos anteriormente. En las tecnologías IoT LTE-M y NB-IoT se realizó una prueba de conectividad

y se inspeccionó el tráfico generado para su análisis junto a una pequeñas comparativa de la latencia y *throughput*.

En relación a estructurar las mediciones en una base de datos y el posterior análisis de los resultados se utilizó el lenguaje de programación Python por su sencillez. Se realizaron pequeños códigos para poder estructurar los archivos obtenidos en la pruebas y otros para identificar los datos guardados en estos documentos y hacer gráficas para una interpretación más rápida y simple. Se optó por gráficas CDF. En LTE-M y NB-IoT se realizó un análisis mediante Wireshark y con la herramienta Web de Amarisoft para estudiar los mensajes tanto de conexión como los enviados. Además, se hizo un pequeño análisis del *throughput* y latencia, que no se tenía planteado inicialmente en los objetivos del proyecto.

Por último, al analizar los datos se ha concluido que con las pruebas realizadas la red de 5G-SA presentan mejores resultados de rendimiento llegando a picos de 450 Mbits/seg. En cuanto a latencia, es también la tecnología que consigue un menor tiempo, llegando a estar alrededor de 10 ms. Otros resultados a destacar es que únicamente para 5G-SA se obtienen mejores resultados tanto de *throughput* como latencia al utilizar la duplexación de TDD. Fue un éxito total la prueba de conectividad de las tecnologías LTE-M y NB-IoT utilizando sockets entre una estación base y dos UEs en el mismo instante de tiempo. Destacar que LTE-M consigue una menor latencia con un mayor *throughput* y de una forma más estable que NB-IoT.

7.2. Líneas futuras

Los objetivos previsto se han llevado a cabo, pero hay muchas mejoras que se pueden realizar así como línea futuras para seguir con este proyecto, algunas se citan a continuación:

- Desplegar y realizar las mismas pruebas pero con otra de las alternativas de mencionadas en la sección del Estado del arte 2 como 5GS.
- Investigar e implementar que dispositivos para que actúen de UEs soportan un *subcarrier spacing* de 30 KHz para FDD y de 60 KHz para ambas duplexaciones. Ya que el software de Amarisoft tolera esto pero los Ues con los que disponemos actualmente (Tarjeta Quectel) únicamente soporta 15 KHz para FDD y hasta 30 KHz para TDD.
- Investigar y probar N3IWF para implementar Wi-Fi integrándola en la red de 5G.
- Investigar network slicing en las redes de 5G, ya que en el proceso del proyecto se intentaron varias pruebas de esta técnica pero sin un éxito

total. Por lo tanto, es de gran interés implementar estas técnicas para dar una mayor flexibilidad y calidad de servicio. Abriendo un gran abanico de posibilidades a la hora de realizar pruebas.

Bibliografía

[1] *Data Never Sleeps*. Disponible en: https://www.domo.com/data-never-sleeps (último acceso 13/03/2023).

[2] *El despliegue de la 5G en el mundo*. Disponible en: https://es.statista.com/grafico/23241/nivel-de-desarrollo-de-la-tecnologia-5g-en-el-mundo/ (último acceso 13/03/2023).

[3] *5G Subscribers March 2023 update*. Disponible en: https://gsacom.com/paper/5g-subscribers-march-2023-update/ (último acceso 15/03/2023).

[4] *Release 18*. Disponible en: https://www.3gpp.org/specifications-technologies/releases/release-18 (último acceso 15/03/2023).

[5] *Release 13*. Disponible en: https://www.3gpp.org/specifications-technologies/releases/release-13 (último acceso 04/06/2023).

[6] Juan Carlos Cano et al. "Evolution of IoT: an industry perspective". En: *IEEE Internet of Things Magazine* 1.2 (2018), págs. 12-17.

[7] *What is free5GC?* Disponible en: https://www.free5gc.org/ (último acceso 16/03/2023).

[8] *Release 15*. Disponible en: https://www.3gpp.org/specifications-technologies/releases/release-15 (último acceso 03/04/2023).

[9] *Free5GC GitHub*. Disponible en: https://github.com/free5gc/free5gc (último acceso 16/03/2023).

[10] *Free5GC GitHub Wiki*. Disponible en: https://github.com/free5gc/free5gc/wiki (último acceso 17/03/2023).

[11] *Open Source implementation for 5G Core and EPC, i.e. the core network of LTE/NR network (Release-17)*. Disponible en: https://open5gs.org/ (último acceso 17/03/2023).

[12] *Release 17*. Disponible en: https://www.3gpp.org/specifications-technologies/releases/release-17 (último acceso 25/05/2023).

[13] *Open5GS GitHub*. Disponible en: https://github.com/open5gs (último acceso 17/03/2023).

[14] *Quickstart Open5GS*. Disponible en: `https://open5gs.org/open5gs/docs/guide/01-quickstart/` (último acceso 17/03/2023).

[15] *UERANSIM GitHub*. Disponible en: `https://github.com/aligungr/UERANSIM` (último acceso 17/03/2023).

[16] *About the OpenAirInterface Software Alliance*. Disponible en: `https://openairinterface.org/about-us/` (último acceso 19/03/2023).

[17] *Projects OpenAirInterfance*. Disponible en: `https://openairinterface.org/projects/` (último acceso 19/03/2023).

[18] *SrsRAN GitHub*. Disponible en: `https://github.com/srsran` (último acceso 20/03/2023).

[19] *Open Source RAN*. Disponible en: `https://www.srslte.com/` (último acceso 20/03/2023).

[20] *srsRAN 4G 22.10 Documentation*. Disponible en: `https://docs.srsran.com/projects/4g/en/latest/` (último acceso 20/03/2023).

[21] *srsRAN Project Documentation*. Disponible en: `https://docs.srsran.com/projects/project/en/latest/` (último acceso 20/03/2023).

[22] *srsRAN Project GitHub*. Disponible en: `https://github.com/srsran/srsRAN_Project` (último acceso 20/03/2023).

[23] *Amarisoft, About us*. Disponible en: `https://www.amarisoft.com/about-us/` (último acceso 20/03/2023).

[24] *Amarisoft, Products*. Disponible en: `https://www.amarisoft.com/products/` (último acceso 20/03/2023).

[25] *Amarisoft, Technology*. Disponible en: `https://www.amarisoft.com/technology/` (último acceso 21/03/2023).

[26] *Amarisoft, AMARI Callbox Series*. Disponible en: `https://www.amarisoft.com/products/test-measurements/amari-lte-callbox/` (último acceso 21/03/2023).

[27] *Release 8*. Disponible en: `https://www.3gpp.org/specifications-technologies/releases/release-8` (último acceso 04/06/2023).

[28] *Releases Overview*. Disponible en: `https://www.3gpp.org/specifications-technologies/releases` (último acceso 29/03/2023).

[29] *Release 10*. Disponible en: `https://www.3gpp.org/specifications-technologies/releases/release-10` (último acceso 04/06/2023).

[30] Magri Hicham, Noreddine Abghour y Mohammed Ouzzif. "4G system: network architecture and performance". En: *International Journal of Innovative Research Advanced Engineering (IJIRAE)* 2.4 (2015), págs. 215-220.

[31] Ian F Akyildiz et al. "LTE-Advanced and the evolution to Beyond 4G (B4G) systems". En: *Physical Communication* 10 (2014), págs. 31-60.

[32] Harri Holma y Antti Toskala. *LTE for UMTS: Evolution to LTE-advanced*. John Wiley & Sons, 2011.

[33] Gabriel Maciá Fernández, Roberto Magán Carrión y Rafael A. Rodríguez Gómez. *Sistemas y servicios telemáticos*. Editorial Técnica AVICAM, 2018.

[34] Stefania Sesia, Issam Toufik y Matthew Baker. *LTE-The UMTS Long Term Evolution: From Theory to Practice*. John Wiley & Sons, 2011.

[35] Erik Dahlman, Stefan Parkvall y Johan Sköld. *4G: LTE/LTE-Advanced for Mobile Broadband*. Academic Press, 2013.

[36] *Protocolos E-UTRAN*. Disponible en: `http://www.ipv6go.net/lte/eutran_protocolos.php` (último acceso 31/03/2023).

[37] *Entendiendo la diferencia entre FDD vs TDD en sistema de microondas*. Disponible en: `https://aviatnetworks.com/blog/diferencia-entre-fdd-vs-tdd` (último acceso 31/03/2023).

[38] *4G LTE Frequency Bands*. Disponible en: `https://www.sqimway.com/lte_band.php` (último acceso 03/04/2023).

[39] Erik Dahlman, Stefan Parkvall y Johan Sköld. *5G NR: The Next Generation Wireless Access Technology*. Academic Press, 2020.

[40] *5G System Overview*. Disponible en: `https://www.3gpp.org/technologies/5g-system-overview` (último acceso 03/04/2023).

[41] Sassan Ahmadi. *5G NR: Architecture, Technology, Implementation, and Operation of 3GPP New Radio Standards*. Academic Press, 2018.

[42] *La pirámide de 5G: Servicios eMBB, URLLC y mMTC*. Disponible en: `https://developer.qualcomm.com/blog/enhanced-mobile-broadband-5g-innovation-consumers` (último acceso 05/07/2023).

[43] *5G; System architecture for the 5G System (5GS)*. Disponible en: `https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.08.00_60/ts_123501v160800p.pdf` (último acceso 05/04/2023).

[44] *5G NR Frequency Band*. Disponible en: `https://www.sqimway.com/nr_band.phpACC` (último acceso 10/04/2023).

[45] Rapeepat Ratasuk et al. "LTE-M evolution towards 5G massive MTC". En: *2017 IEEE Globecom Workshops (GC Wkshps)*. IEEE. 2017, págs. 1-6.

[46] Khaled Aldahdouh Aldahdouh, Khalid A Darabkh, Waleed Al-Sit et al. "A survey of 5G emerging wireless technologies featuring LoRaWAN, Sigfox, NB-IoT and LTE-M". En: *2019 International conference on wireless communications signal processing and networking (WiSPNET)*. IEEE. 2019, págs. 561-566.

[47] Almudena Dıaz Zayas, Pedro Merino Gómez y Francisco Rivas Tocado. "3GPP NB-IoT, tecnologıa y herramientas de medida". En: *XIII Jornadas de Ingenieria telemática (JITEL 2017). Libro de actas* (2018), págs. 310-317.

[48] Rapeepat Ratasuk et al. "NB-IoT system for M2M communication". En: *2016 IEEE wireless communications and networking conference*. IEEE. 2016, págs. 1-5.

[49] *4G/LTE - NB IoT*. Disponible en: `https://www.sharetechnote.com/html/Handbook_LTE_NB_LTE.html` (último acceso 21/06/2023).

[50] *Release 14*. Disponible en: `https://www.3gpp.org/specifications-technologies/releases/release-14` (último acceso 25/06/2023).

[51] *NB-IoT Frequency Band*. Disponible en: `https://www.cablefree.net/wirelesstechnology/internet-of-things-iot/` (último acceso 25/06/2023).

[52] *ASUS Zenbook 13 UX325JA*. Disponible en: `https://www.asus.com/mx/laptops/for-home/zenbook/asus-zenbook-13-ux325ja/techspec/` (último acceso 01/06/2023).

[53] *5G RM50xQ series*. Disponible en: `https://www.quectel.com/product/5g-rm50xq-series` (último acceso 31/05/2023).

[54] *FiPy - Datasheet*. Disponible en: `https://pycom.io/wp-content/uploads/2018/03/Pycom_002_Specsheets_FiPy_v2.pdf` (último acceso 31/05/2023).

[55] *Visual Studio Code. Code editing. Redefined*. Disponible en: `https://code.visualstudio.com/` (último acceso 31/05/2023).

[56] *Página oficial de MobaXterm,* Disponible en: `https://mobaxterm.mobatek.net/` (último acceso 25/06/2023).

[57] *Github oficial de Pysim*. Disponible en: `https://github.com/osmocom/pysim` (último acceso 03/06/2023).

[58] *Download Wireshark*. Disponible en: `https://www.wireshark.org/download.html` (último acceso 31/05/2023).

[59] *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*. Disponible en: `https://iperf.fr/iperf-download.php` (último acceso 01/06/2023).

[60] *Overleaf - The LaTeX solution for everyone*. Disponible en: `https://es.overleaf.com/` (último acceso 10/07/2023).

[61] *Amarisoft - Screen User's Manual*. Disponible en: `https://www.gnu.org/software/screen/manual/screen.html` (último acceso 16/06/2023).

[62] Vibhuti Narayan Singh, Shalini Chauhan y G Khan. "Forensic analysis of SIM cards for data acquisition". En: *Asian Journal of Multidisciplinary Studies* 3.1 (2015), pág. 25.

[63] Fabian Van Den Broek, Roel Verdult y Joeri De Ruiter. "Defeating IMSI catchers". En: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, págs. 340-351.

[64] *Official page of Busybox*. Disponible en: `https://busybox.net/` (último acceso 16/06/2023).

[65] Panagiotis Tzimotoudis, Nikos Makris y Thanasis Korakis. "Design and Implementation of an Open-Source NB-IoT Core Network". En: *Proceedings of the 11th International Conference on the Internet of Things*. 2021, págs. 200-202.

Glosario

3G Tercera generación de tecnología móvil.

3GPP Third Generation Partnership Project.

4G Cuarta generación de tecnología móvil.

5G Quinta generación de tecnología móvil.

5GC 5G Core.

ADM1 Administrative Key.

AF Access Anchor Function.

AMF Access and Mobility Management Function.

APN Access Point Name .

ARFCN Absolute Radio Frequency Channel Number .

AUSF Authentication Server Function.

BSC Base Station Controller.

CAT-M1 Category M1.

CDF Cumulative Distribution Function .

CN Core Network.

CU Control User.

DU Distributed User.

E-UTRAN Enhanced UMTS Terrestrial Radio Access Network.

eDRX Extended Discontinuous Reception.

eMBB Enhance mobile broadband.

EMEA Europe the Middle East and Africa.

eNB Evolved Node B.

EPC Evolved Packet Core.

ESM EPS Session Management .

EU European Union.

FDD Frequency Division Duplex.

FDM Frequency Division Multiplexing.

gNB gNodeB.

GSA Global Mobile Suppliers Association.

GSM Global System For Mobile Comunication.

GTP GPRS Tunnelling Protocol.

GUI Graphical User Interface.

HSS Home Subscriber Server.

ICCID Integrated Circuit Card Identifier.

IMPI IP Multimedia Private Identity.

IMPU IP Multimedia Public Identity.

IMS IP Multimedia Subsystem.

IMSI International Mobile Subscriber Identify.

IoT Internet of Things.

IP Internet Protocol.

IPv4 Internet Protocol version 4.

Ki Authentication key.

LAI Location Area Identity.

LTE Long Term Evolution.

LTE-M Long Term Evolution (4G) categoría M1.

MAC Media Access Control.

MBMSGW Mobile Broadcast Multicast Service Gateway.

MIMO Multiple-Input Multiple-Output.

MME Mobility Management Entity.

mMTC Massive Machine-Type Communications.

N3IWF Non-3GPP Inter Working Function.

NAS Non-Access-Stratum Protocol.

NB-IoT Narrowband IoT.

NEF Network Exposure Function.

NF Network Function.

NGAP Next Generation Application Protocol.

NR New Radio.

NR New Radio.

NRF Network Repository Function.

NSA Non-Stand Alone.

NSSF Network Slice Selection Function.

OFDM Orthogonal Frequency-Division Multiplexing.

OFDMA Orthogonal Frequency Division Multiple Access.

OSA OpenAirInterface Software Alliance.

PCF Policy Control Function.

PDCP Packet Data Convergence Protocol.

PGW Packet Data Network Gateway.

PLMN Public Land Mobile Network.

QPSK Quadrature Phase Shift Keying.

RAN Radio Access Network.

RB Resource Block.

RF Radio Frequency.

RLC Radio Link Control.

RNC Radio Network Controller.

RRC Radio Resource Control.

RTT Round Trip Time .

S-GW Serving Gateway.

SA Stand-Alone.

SBA Service Based Architecture.

SC-FDMA Single Carrier Frequency Division Multiple Access.

SDR Software Defined Radio.

SISO Single-output single-input.

SMF Session Management Function.

SON Self-Organizing Networks.

SRS Software Radio Systems.

SSH Secure SHell .

SUPI Subscription Permanent Identifier.

TCP Transmission Control Protocol.

TDD Time Division Duplex.

TDM Time Division Multiplexing.

UDM Unified Data Manager.

UDP User Datagram protocol.

UDR Unified Data Repository.

UE User Equipment.

ULCL UP Uplink Classifier.

UPF User Plane Function.

URLLC Ultra-High Reliability & Low Latency.

USB Universal Serial Bus.

USIM User Subscriber Identification Module.

Wi-Fi Wireless Fidelity.

Apéndice A

Automatización de la conexión de la tarjeta quectel

connect_quectel_amarisoft.sh

```
#!/bin/bash

# APN ('Internet' for Amarisoft, 'srsapn' for srsRAN)
APN='Internet'

# Select whether a namespace ('quectel') will be used or not
USENS=1

#INTERFACE='wwp0s20f0u3i4'
INTERFACE=`ip a | grep wwp | cut -d" " -f2 | sed 's/:*$//g'`

if [[ $INTERFACE == "" ]]; then
  echo "No Quectel card (interface wwp*) found! Exiting..."
  exit
else
  echo "Found Quectel card! Network interface is ${INTERFACE}"
fi

if [[ $USENS == 1 ]]; then
  # Add a namespace so it can be used in the PC with the base station
  TESTNS=`sudo ip netns ls | grep quectel | cut -d" " -f1`
  if [[ $TESTNS == 'quectel' ]]; then
    echo "Namespace quectel already exists!"
  else
    echo "Creating namespace quectel"
    sudo ip netns add quectel
  fi

  # Move the network interface of the Quectel card to the namespace
  TESTIF=`ip a | grep $INTERFACE | cut -d" " -f2 | sed s/://g`
  if [[ $TESTIF == $INTERFACE ]]; then
    echo "Moving interface ${INTERFACE} to quectel namespace"
    sudo ip link set ${INTERFACE} netns quectel
  else
    TESTIF=`sudo ip netns exec quectel ip a | grep $INTERFACE | cut -d
        " " -f2 | sed s/://g`
    if [[ $TESTIF == $INTERFACE ]]; then
```

```
37        echo "Interface ${INTERFACE} is already in quectel namespace!"
38      else
39        echo "Interface ${INTERFACE} does not exist in the global or in
              quectel namespaces. Exiting..."
40        exit
41      fi
42    fi
43
44    # Connects to the mobile network (APN name: $APN; the rest of the
        parameters are their default values (e.g. APN type is 'default')
        )
45    sudo ip netns exec quectel quectel-CM -s $APN
46
47  else
48
49    # Connects to the mobile network (APN name: $APN; the rest of the
        parameters are their default values (e.g. APN type is 'default')
        )
50    sudo quectel-CM -s $APN
51
52  fi
```

Apéndice B

Automatización de la configuración de las tarjetas USIM

prog_usims_amarisoft_v3.sh

```
ICCID=`./pySim-read.py -p 0 | grep ICCID | cut -b 8-26`
echo "ICCID: ${ICCID}"
MCC="170"
MNC="90"
#OP="504f20634f6320504f50206363500a4f"
OPc="000102030405060708090A0B0C0D0E0F"
#AMF="0x9001"
#key="68747369692020737796D4B2079650A73"
key="00112233445566778899AABBCCDDEEFF"
#pinAMD="11111111"
ADM1=""

if [[ "${ICCID}" == "8988211000000142102" ]]; then
   ADM1="85601364"
   IMSI="208920000000001"
elif [[ "${ICCID}" == "8988211000000142110" ]]; then
   ADM1="10022674"
   IMSI="208920000000002"
elif [[ "${ICCID}" == "8988211000000142128" ]]; then
   ADM1="00415661"
   IMSI="001010000000001"
elif [[ "${ICCID}" == "8988211000000142136" ]]; then
   ADM1="53650514"
   IMSI="208920000000004"
elif [[ "${ICCID}" == "8988211000000142144" ]]; then
   ADM1="04521585"
   IMSI="901700000000001"
elif [[ "${ICCID}" == "8988211000000142151" ]]; then
   ADM1="95681488"
   IMSI="208920000000006"
elif [[ "${ICCID}" == "8988211000000142169" ]]; then
   ADM1="16325989"
   IMSI="208920000000007"
```

```
34 elif [[ "${ICCID}" == "8988211000000142177" ]]; then
35    ADM1="47387169"
36    IMSI="208920000000008"
37 elif [[ "${ICCID}" == "8988211000000142185" ]]; then
38    ADM1="54073191"
39    IMSI="901700000000003"
40 elif [[ "${ICCID}" == "8988211000000142193" ]]; then
41    ADM1="55002353"
42    IMSI="901700000000002"
43 else
44    echo "ICCID not known!"
45 fi
46
47 if [[ "${ADM1}" != "" ]]; then
48 # Cambio a Algo Milenage
49    cvar="000102030405060708090A0B0C0D0E0F"
50    echo "./sysmo-usim-tool.sjs1.py -a ${ADM1} -J ${IMSI} -T MILENAGE:
          MILENAGE -k ${key} -C ${cvar}"
51    ./sysmo-usim-tool.sjs1.py -I ${ICCID} -u -a ${ADM1} -J ${IMSI} -T
          MILENAGE:MILENAGE -k ${key} -C ${cvar}
52
53 # Cambio a algoritmo XOR
54   # echo "./sysmo-isim-tool.sja2.py -a ${ADM1} -J ${IMSI} -T XOR:XOR -
       k ${key}"
55   # ./sysmo-isim-tool.sja2.py -a ${ADM1} -J ${IMSI} -T XOR:XOR -k ${
       key}
56
57
58 fi
```

Apéndice C

Automatización para guardar los datos

perform_test.sh

```
#!/bin/bash

#########################
# Parse inputs parameters
#########################

# Default values
REPETITIONS=15
SERVER="192.168.3.2"
DUPLEX="TDD"
BW="20"
N_ANTENA="MIMO_Prueba"

TEST_NAME="5G_SA_${DUPLEX}_${BW}_${PER}_${N_ANTENA}"

# Perform test
echo "SERVER: ${SERVER}"
echo "REPS: ${REPETITIONS}"
echo "Test ${TEST_NAME}"

# Perform a number of iperf test repetitions
echo "Start Experiment"
for contador in $(seq 1 1 ${REPETITIONS})
do
   echo "Starting test $contador"
   iperf3 -c ${SERVER} -P 10 >> "./Tests5GSA/iperf_${TEST_NAME}.txt"
   sleep 3

   ping ${SERVER} -c 15 >> "./Tests5GSA/ping_${TEST_NAME}.txt"
   sleep 3

done

echo "Test finished"
```

Apéndice D

Códigos de parseo de los datos

parseo_iperf.py

```
import os

def limpiarEspacios(cadena):
    cadena_sin_espacios = cadena.split()
    return cadena_sin_espacios


def parserDocsNew():
    directorio = os.listdir("C:/Users/javij/OneDrive/Escritorio/
        Teleco4/2Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Tests/")
    nameParsed = []

    nDocs = 0
    nameParsed.append("iperf_parsed.csv")
    for fil in directorio:
        if fil.startswith("iperf"):
            nDocs= nDocs + 1
            parserIperf(fil)

    return nDocs, nameParsed

def parserIperf(f):
    directorio = "C:/Users/javij/OneDrive/Escritorio/Teleco4/2Cuatri/
        TFG/Resultados/JITEL5G/TFG_Javi/Tests/"
    fin = open(directorio + f,"r")
    ext=f.split(".")
    fout_iperf = open("C:/Users/javij/OneDrive/Escritorio/Teleco4/2
        Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2/iperf_parsed.
        csv","a")

    try:
        lineaCSV = "Tecnologia,Duplex,BW,MIMO,Interv,Transf,Throughput
            ,PcktRetr"

        if os.path.getsize("C:/Users/javij/OneDrive/Escritorio/Teleco4
            /2Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2/
            iperf_parsed.csv") == 0:
```

```
            fout_iperf.write(lineaCSV + "\n")

        count = 1
        for linea in fin:
            if linea.startswith("[SUM]") and count != 1:
                palabras = limpiarEspacios(linea)

                variables = ext[0].split("_")

                # Formato CSV para facilitar el analisis
                if palabras[1] != "0.00-10.00" :
                    info1 = palabras[1] + "," + palabras[3] + "," +
                        palabras[5] + "," + palabras[7]

                    lineaCSV = variables[1] + "," + variables[2] + ","
                         + variables[3] + "," + variables[4] + "," +
                        info1
                    print(lineaCSV)
                    fout_iperf.write(lineaCSV + "\n")

            count = 2

        fin.close()
        fout_iperf.close()
    except Exception as e:
        print(type(e))
        fin.close()
        fout_iperf.close()


nDocs_retorno, nombre_parseado_retorno = parserDocsNew()

# Imprimir los valores obtenidos
print("numero de documentos:", nDocs_retorno)
print("Nombre parseado:", nombre_parseado_retorno)
```

parseo_ping.py

```
import os

def limpiarEspacios(cadena):
    cadena_sin_espacios = cadena.replace(" ", ",")
    return cadena_sin_espacios

def parserDocsNew():
    directorio = os.listdir("C:/Users/javij/OneDrive/Escritorio/
        Teleco4/2Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Tests/")
    nameParsed = []

    nDocs = 0
    nameParsed.append("rtt_parsed_5GNSA1.csv")
    for fil in directorio:
        if fil.startswith("ping"):
            nDocs= nDocs + 1
            parserRTTNew(fil)

    return nDocs, nameParsed


```

```
21 def parserRTTNew(f):
22     directorio = "C:/Users/javij/OneDrive/Escritorio/Teleco4/2Cuatri/
           TFG/Resultados/JITEL5G/TFG_Javi/Tests/"
23     fin = open(directorio + f,"r")
24     ext=f.split(".")
25 
26     fout_rtt =open("C:/Users/javij/OneDrive/Escritorio/Teleco4/2Cuatri
           /TFG/Resultados/JITEL5G/TFG_Javi/Parsed2/rtt_parsed.csv","a")
27 
28     try:
29 
30         lineaCSV = "Tecnologia,Duplex,BW,MIMO,Ptx,Prx,Pl,Time,RTTmin,
               RTTavg,RTTmax,RTTmdev"
31 
32         # Comprobar si existe el archivo
33         if os.path.getsize("C:/Users/javij/OneDrive/Escritorio/Teleco4
               /2Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2/
               rtt_parsed.csv") == 0:
34             fout_rtt.write(lineaCSV + "\n")
35 
36         variables = ext[0].split("_")
37 
38         for linea in fin:
39 
40             if linea.startswith("15"):
41                 linea= limpiarEspacios(linea)
42                 palabras = linea.split(",")
43                 info1 = palabras[0] + "," + palabras[4] + "," +
                       palabras[7].replace("%","") + "," + palabras[12].
                       replace("ms","").strip()
44                 lineaCSV = variables[1] + "," + variables[2] + "," +
                       variables[3] + "," + variables[4] + "," + info1
45 
46             elif linea.startswith("rtt"):
47                 linea= limpiarEspacios(linea)
48                 palabras = linea.replace(",ms","").replace("rtt,","").
                       replace(",","").split("=")
49                 palabras = palabras[1].split("/")
50                 info2 = palabras[0] + "," + palabras[1] + "," +
                       palabras[2] + "," + palabras[3]
51 
52                 # Formato CSV para facilitar el analisis
53                 lineaCSV +=","+ info2
54                 print(lineaCSV)
55                 fout_rtt.write(lineaCSV)
56 
57 
58         fin.close()
59         fout_rtt.close()
60     except Exception as e:
61         print(type(e))
62         fin.close()
63         fout_rtt.close()
64 
65 nDocs_retorno, nombre_parseado_retorno = parserDocsNew()
66 
67 # Imprimir los valores obtenidos
68 print("numero de documentos:", nDocs_retorno)
69 print("Nombre parseado:", nombre_parseado_retorno)
```

Apéndice E

Códigos para el analisis de los datos mediante gráficas CDF

CDF_Throughput.py

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


# Leer los datos del archivo iperf y almacena en un data frame
iperf_data = pd.read_csv('C:/Users/javij/OneDrive/Escritorio/Teleco4/2 Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2-Prueba/iperf_parsed.csv')

###---------------------------Tipo de tecnologia ------------------------
df_filtrado4G= iperf_data[iperf_data['Tecnologia'] == '4G']
df_filtrado5GNSA= iperf_data[iperf_data['Tecnologia'] == '5GNSA']
df_filtrado5GSA= iperf_data[iperf_data['Tecnologia'] == '5GSA']


# tecnologia a considerar
tecnologia_vector=['4G', '5GNSA', '5GSA']

## CDF para diferentes tecnologias dependiendo del Transf

for tecnologia in tecnologia_vector:

    df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]

    df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y remplaza en el df_filtrado

    df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado)


```

```
29      plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label=
            tecnologia)
30
31
32  plt.xlabel('Transf')
33  plt.ylabel('CDF')
34  plt.title('Comparativa de la transferencia para las tecnologias ')
35  plt.grid(True)
36  plt.legend()
37  plt.show()
38
39  ###---------------------------Numero antenas-----------------------
40
41  ## CDF para 4G depende de MIMO
42  antenas_vector =['SISO', 'MIMO']
43
44  for antena in antenas_vector:
45
46      df_filtrado = df_filtrado4G[df_filtrado4G['MIMO'] == antena]
47
48      df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
            remplaza en el df_filtrado
49
50      df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
51
52
53      plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= antena)
54
55
56  plt.xlabel('Transf')
57  plt.ylabel('CDF')
58  plt.title('CDF de 4G dependiendo del numero antenas')
59  plt.grid(True)
60  plt.legend()
61  plt.show()
62
63  ## CDF para 5GNSA depende de MIMO
64
65  for antena in antenas_vector:
66
67
68      df_filtrado = df_filtrado5GNSA[df_filtrado5GNSA['MIMO'] == antena]
69
70      df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
            remplaza en el df_filtrado
71
72      df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
73
74
75      plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= antena)
76
77
78  plt.xlabel('Transf')
79  plt.ylabel('CDF')
80  plt.title('CDF de 5GNSA dependiendo del numero antenas')
81  plt.grid(True)
82  plt.legend()
83  plt.show()
84
85  ## CDF para 5GSA depende de MIMO
```

```
86
87 for antena in antenas_vector:
88
89     df_filtrado = df_filtrado5GSA[df_filtrado5GSA['MIMO'] == antena]
90
91     df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
           remplaza en el df_filtrado
92
93     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
           )
94
95
96     plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= antena)
97
98
99 plt.xlabel('Transf')
100 plt.ylabel('CDF')
101 plt.title('CDF de 5GSA dependiendo del numero antenas')
102 plt.grid(True)
103 plt.legend()
104 plt.show()
105
106
107  ###--------------------------DUPLEX-----------------------
108
109 ## CDF para 4G depende de DUPLEX
110 duplex_vector =['FDD', 'TDD']
111
112 for duplex in duplex_vector:
113
114
115     df_filtrado = df_filtrado4G[df_filtrado4G['Duplex'] == duplex]
116
117     df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
           remplaza en el df_filtrado
118
119     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
           )
120
121
122     plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= duplex)
123
124
125 plt.xlabel('Transf')
126 plt.ylabel('CDF')
127 plt.title('CDF de 4G dependiendo de la duplexacion')
128 plt.grid(True)
129 plt.legend()
130 plt.show()
131
132 ## CDF para 5GNSA depende de DUPLEX
133
134 for duplex in duplex_vector:
135
136
137     df_filtrado = df_filtrado5GNSA[df_filtrado5GNSA['Duplex'] ==
           duplex]
138
139     df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
           remplaza en el df_filtrado
140
```

```
141     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
142
143
144     plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= duplex)
145
146
147 plt.xlabel('Transf')
148 plt.ylabel('CDF')
149 plt.title('CDF de 5GNSA dependiendo de la duplexacion')
150 plt.grid(True)
151 plt.legend()
152 plt.show()
153
154 ## CDF para 5GNSA depende de DUPLEX
155
156 for duplex in duplex_vector:
157
158
159     df_filtrado = df_filtrado5GSA[df_filtrado5GSA['Duplex'] == duplex]
160
161     df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
            remplaza en el df_filtrado
162
163     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
164
165
166     plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= duplex)
167
168
169 plt.xlabel('Transf')
170 plt.ylabel('CDF')
171 plt.title('CDF de 5GSA dependiendo de la duplexacion')
172 plt.grid(True)
173 plt.legend()
174 plt.show()
175
176
177 ###---------------------------BW-----------------------
178
179 ## CDF para 4G depende de DUPLEX
180 BW_vector =[5, 10, 15, 20]
181
182 for BW in BW_vector:
183
184     df_filtrado = df_filtrado4G[df_filtrado4G['BW'] == BW]
185
186     df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
            remplaza en el df_filtrado
187
188     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
189
190
191     plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= BW)
192
193
194 plt.xlabel('Transf')
195 plt.ylabel('CDF')
196 plt.title('CDF de 4G dependiendo del ancho de banda')
197 plt.grid(True)
```

```
plt.legend()
plt.show()

## CDF para 5GNSA depende de DUPLEX

for BW in BW_vector:

    df_filtrado = df_filtrado5GNSA[df_filtrado5GNSA['BW'] == BW]

    df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
        remplaza en el df_filtrado

    df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
        )


    plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= BW)


plt.xlabel('Transf')
plt.ylabel('CDF')
plt.title('CDF de 5GNSA dependiendo del ancho de banda')
plt.grid(True)
plt.legend()
plt.show()

## CDF para 5GSA depende de DUPLEX
BW_vector5GSA =[5, 10, 15, 20, 40, 50]

for BW in BW_vector5GSA:

    df_filtrado = df_filtrado5GSA[df_filtrado5GSA['BW'] == BW]

    df_filtrado.sort_values(by='Transf', inplace=True) #Ordena y
        remplaza en el df_filtrado

    df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
        )


    plt.plot(df_filtrado['Transf'], df_filtrado['CDF'], label= BW)


plt.xlabel('Transf')
plt.ylabel('CDF')
plt.title('CDF de 5GSA dependiendo del ancho de banda')
plt.grid(True)
plt.legend()
plt.show()
```

CDF_Latencia.py

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


# Leer los datos del archivo iperf y almacena en un data frame
```

```
iperf_data = pd.read_csv('C:/Users/javij/OneDrive/Escritorio/Teleco4/2 Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2-Prueba/rtt_parsed_3.csv')

df_filtrado4G= iperf_data[iperf_data['Tecnologia'] == '4G']
df_filtrado5GNSA= iperf_data[iperf_data['Tecnologia'] == '5GNSA']
df_filtrado5GSA= iperf_data[iperf_data['Tecnologia'] == '5GSA']

# tecnologia a considerar
tecnologia_vector=['4G', '5GNSA', '5GSA']

## CDF para diferentes tecnologias dependiendo del Latencia

for tecnologia in tecnologia_vector:

    df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]

    df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y remplaza en el df_filtrado

    df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)

    df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado)


    plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label=tecnologia)


plt.xlabel('Latencia')
plt.ylabel('CDF')
plt.title('Comparativa de latencia para las tecnologias ')
plt.grid(True)
plt.legend()
plt.show()

###--------------------------Numero antenas-----------------------

## CDF para 4G depende de MIMO
antenas_vector =['SISO', 'MIMO']

for antena in antenas_vector:

    df_filtrado = df_filtrado4G[df_filtrado4G['MIMO'] == antena]

    df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y remplaza en el df_filtrado

    df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)

    df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado)


    plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label=antena)


plt.xlabel('Latencia')
plt.ylabel('CDF')
plt.title('CDF de 4G dependiendo del numero antenas')
```

```
plt.grid(True)
plt.legend()
plt.show()

## CDF para 5GNSA depende de MIMO

for antena in antenas_vector:


    df_filtrado = df_filtrado5GNSA[df_filtrado5GNSA['MIMO'] == antena]

    df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
        remplaza en el df_filtrado

    df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)

    df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
        )


    plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label=
        antena)


plt.xlabel('Latencia')
plt.ylabel('CDF')
plt.title('CDF de 5GNSA dependiendo del numero antenas')
plt.grid(True)
plt.legend()
plt.show()

## CDF para 5GSA depende de MIMO

for antena in antenas_vector:

    df_filtrado = df_filtrado5GSA[df_filtrado5GSA['MIMO'] == antena]

    df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
        remplaza en el df_filtrado

    df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)

    df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
        )


    plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label=
        antena)


plt.xlabel('Latencia')
plt.ylabel('CDF')
plt.title('CDF de 5GSA dependiendo del numero antenas')
plt.grid(True)
plt.legend()
plt.show()


 ###--------------------------DUPLEX-----------------------

## CDF para 4G depende de DUPLEX
duplex_vector =['FDD', 'TDD']
```

```
117
118 for duplex in duplex_vector:
119
120
121     df_filtrado = df_filtrado4G[df_filtrado4G['Duplex'] == duplex]
122
123     df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
        remplaza en el df_filtrado
124
125     df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)
126
127     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
        )
128
129
130     plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label=
        duplex)
131
132
133 plt.xlabel('Latencia')
134 plt.ylabel('CDF')
135 plt.title('CDF de 4G dependiendo de la duplexacion')
136 plt.grid(True)
137 plt.legend()
138 plt.show()
139
140 ## CDF para 5GNSA depende de DUPLEX
141
142 for duplex in duplex_vector:
143
144
145     df_filtrado = df_filtrado5GNSA[df_filtrado5GNSA['Duplex'] ==
        duplex]
146
147     df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
        remplaza en el df_filtrado
148
149     df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)
150
151     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
        )
152
153
154     plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label=
        duplex)
155
156
157 plt.xlabel('Latencia')
158 plt.ylabel('CDF')
159 plt.title('CDF de 5GNSA dependiendo de la duplexacion')
160 plt.grid(True)
161 plt.legend()
162 plt.show()
163
164 ## CDF para 5GNSA depende de DUPLEX
165
166 for duplex in duplex_vector:
167
168
169     df_filtrado = df_filtrado5GSA[df_filtrado5GSA['Duplex'] == duplex]
170
```

```
171     df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
            remplaza en el df_filtrado
172
173     df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)
174
175     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
176
177
178     plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label=
            duplex)
179
180
181 plt.xlabel('Latencia')
182 plt.ylabel('CDF')
183 plt.title('CDF de 5GSA dependiendo de la duplexacion')
184 plt.grid(True)
185 plt.legend()
186 plt.show()
187
188
189 ###---------------------------BW-------------------------
190
191 ## CDF para 4G depende de DUPLEX
192 BW_vector =[5, 10, 15, 20]
193
194 for BW in BW_vector:
195
196     df_filtrado = df_filtrado4G[df_filtrado4G['BW'] == BW]
197
198     df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
            remplaza en el df_filtrado
199
200     df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)
201
202     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
203
204
205     plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label= BW)
206
207
208 plt.xlabel('Latencia')
209 plt.ylabel('CDF')
210 plt.title('CDF de 4G dependiendo del ancho de banda')
211 plt.grid(True)
212 plt.legend()
213 plt.show()
214
215 ## CDF para 5GNSA depende de DUPLEX
216
217 for BW in BW_vector:
218
219     df_filtrado = df_filtrado5GNSA[df_filtrado5GNSA['BW'] == BW]
220
221     df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
            remplaza en el df_filtrado
222
223     df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)
224
225     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
```

```
226
227
228     plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label= BW)
229
230
231 plt.xlabel('Latencia')
232 plt.ylabel('CDF')
233 plt.title('CDF de 5GNSA dependiendo del ancho de banda')
234 plt.grid(True)
235 plt.legend()
236 plt.show()
237
238 ## CDF para 5GSA depende de DUPLEX
239 BW_vector5GSA =[5, 10, 15, 20, 40, 50]
240
241 for BW in BW_vector5GSA:
242
243     df_filtrado = df_filtrado5GSA[df_filtrado5GSA['BW'] == BW]
244
245     df_filtrado.sort_values(by='RTTavg', inplace=True) #Ordena y
            remplaza en el df_filtrado
246
247     df_filtrado['Latencia'] = df_filtrado['RTTavg'].div(2)
248
249     df_filtrado['CDF'] = np.arange(len(df_filtrado)) / len(df_filtrado
            )
250
251
252     plt.plot(df_filtrado['Latencia'], df_filtrado['CDF'], label= BW)
253
254
255 plt.xlabel('Latencia')
256 plt.ylabel('CDF')
257 plt.title('CDF de 5GSA dependiendo del ancho de banda')
258 plt.grid(True)
259 plt.legend()
260 plt.show()
```

CDF_Latencia_max_min.py

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  # Leer los datos del archivo iperf y almacena en un data frame
6  iperf_data = pd.read_csv('C:/Users/javij/OneDrive/Escritorio/Teleco4/2
       Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2-Prueba/rtt_parsed_2
       .csv')
7
8  tecnologia_vector=['4G', '5GNSA', '5GSA']
9
10 df_filtrado4G= iperf_data[iperf_data['Tecnologia'] == '4G']
11 df_filtrado5GNSA= iperf_data[iperf_data['Tecnologia'] == '5GNSA']
12 df_filtrado5GSA= iperf_data[iperf_data['Tecnologia'] == '5GSA']
13
14
15 #Comparativa latencia media
16 df_grouped = iperf_data.groupby('Tecnologia')['RTTavg'].mean().
       reset_index()
```

```
df_grouped['RTTavg'] /= 2
# Grafica el promedio del RTTavg para cada tecnologia
plt.bar(df_grouped['Tecnologia'], df_grouped['RTTavg'])
plt.xlabel('Tiempo')
plt.ylabel('RTTavg Promedio')
plt.title('Comparacion de RTTavg por tecnologia')

plt.show()


## CDF comparativa 4G tecnologias max_min
vector_max_min=['RTTmin','RTTavg','RTTmax']
for max_min in vector_max_min:


    df_filtrado4G.sort_values(by=max_min, inplace=True) #Ordena y
        remplaza en el df_filtrado

    df_filtrado4G[max_min] /= 2

    df_filtrado4G['CDF'] = np.arange(len(df_filtrado4G)) / len(
        df_filtrado4G)

    if(max_min=='RTTmin' or max_min=='RTTmax'):
        plt.plot(df_filtrado4G[max_min], df_filtrado4G['CDF'], label=
            max_min, linestyle='--')
    else:
        plt.plot(df_filtrado4G[max_min], df_filtrado4G['CDF'], label=
            max_min)


plt.xlabel('RTTavg')
plt.ylabel('CDF')
plt.title('Comparativa latencia para 4G ')
plt.grid(True)
plt.legend()
plt.show()

## CDF comparativa 5GNSA tecnologias max_min

for max_min in vector_max_min:


    df_filtrado5GNSA.sort_values(by=max_min, inplace=True) #Ordena y
        remplaza en el df_filtrado

    df_filtrado5GNSA[max_min] /= 2

    df_filtrado5GNSA['CDF'] = np.arange(len(df_filtrado5GNSA)) / len(
        df_filtrado5GNSA)

    if(max_min=='RTTmin' or max_min=='RTTmax'):
        plt.plot(df_filtrado5GNSA[max_min], df_filtrado5GNSA['CDF'],
            label= max_min, linestyle='--')
    else:
        plt.plot(df_filtrado5GNSA[max_min], df_filtrado5GNSA['CDF'],
            label= max_min)


plt.xlabel('RTTavg')
plt.ylabel('CDF')
plt.title('Comparativa latencia para 5GNSA ')
```

```
plt.grid(True)
plt.legend()
plt.show()

## CDF comparativa 5GSA tecnologias max_min

for max_min in vector_max_min:


    df_filtrado5GSA.sort_values(by=max_min, inplace=True) #Ordena y
        remplaza en el df_filtrado

    df_filtrado5GSA[max_min] /= 2

    df_filtrado5GSA['CDF'] = np.arange(len(df_filtrado5GSA)) / len(
        df_filtrado5GSA)

    if(max_min=='RTTmin' or max_min=='RTTmax'):
        plt.plot(df_filtrado5GSA[max_min], df_filtrado5GSA['CDF'],
            label= max_min, linestyle='--')
    else:
        plt.plot(df_filtrado5GSA[max_min], df_filtrado5GSA['CDF'],
            label= max_min)


plt.xlabel('RTTavg')
plt.ylabel('CDF')
plt.title('Comparativa latencia para 5GSA ')
plt.grid(True)
plt.legend()
plt.show()
```

CDF_Throughput_all.py

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


# Leer los datos del archivo iperf y almacena en un data frame
iperf_data = pd.read_csv('C:/Users/javij/OneDrive/Escritorio/Teleco4/2
    Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2-Prueba/iperf_parsed
    .csv')

tecnologia_vector=['4G', '5GNSA', '5GSA']


## CDF depende de MIMO
antenas_vector =['SISO', 'MIMO']
for tecnologia in tecnologia_vector:
    df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]
    for antena in antenas_vector:

        df_filtrado2 = df_filtrado[df_filtrado['MIMO'] == antena]

        df_filtrado2.sort_values(by='Throughput', inplace=True) #
            Ordena y remplaza en el df_filtrado

```

```
22          df_filtrado2['CDF'] = np.arange(len(df_filtrado2)) / len(
                df_filtrado2)
23
24
25          plt.plot(df_filtrado2['Throughput'], df_filtrado2['CDF'],
                label= tecnologia + " " + antena)
26
27
28  plt.xlabel('Throughput')
29  plt.ylabel('CDF')
30  plt.title('CDF Throughput dependiendo del numero antenas')
31  plt.grid(True)
32  plt.legend()
33  plt.show()
34
35  ### CDF depende de DUPLEX
36  duplex_vector =['FDD', 'TDD']
37  for tecnologia in tecnologia_vector:
38      df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]
39      for duplex in duplex_vector:
40
41
42          df_filtrado2 = df_filtrado[df_filtrado['Duplex'] == duplex]
43
44          df_filtrado2.sort_values(by='Throughput', inplace=True) #
                Ordena y remplaza en el df_filtrado
45
46          df_filtrado2['CDF'] = np.arange(len(df_filtrado2)) / len(
                df_filtrado2)
47
48
49          plt.plot(df_filtrado2['Throughput'], df_filtrado2['CDF'],
                label= tecnologia + " " + duplex)
50
51
52  plt.xlabel('Throughput')
53  plt.ylabel('CDF')
54  plt.title('CDF Throughput dependiendo de la duplexacion')
55  plt.grid(True)
56  plt.legend()
57  plt.show()
58
59  ## CDF depende de BW
60  BW_vector =[5, 10, 15, 20, 40, 50]
61  for tecnologia in tecnologia_vector:
62      df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]
63
64      if tecnologia == '5GSA':
65
66          BW_range = BW_vector
67      else:
68
69          BW_range = BW_vector[:4]
70
71      for BW in BW_range:
72
73
74          df_filtrado2 = df_filtrado[df_filtrado['BW'] == BW]
75
76          df_filtrado2.sort_values(by='Throughput', inplace=True) #
                Ordena y remplaza en el df_filtrado
77
```

```
        df_filtrado2['CDF'] = np.arange(len(df_filtrado2)) / len(
            df_filtrado2)


        plt.plot(df_filtrado2['Throughput'], df_filtrado2['CDF'],
            label=tecnologia + " " + str(BW)+"MHz")


plt.xlabel('Throughput')
plt.ylabel('CDF')
plt.title('CDF Throughput dependiendo del ancho de banda')
plt.grid(True)
plt.legend()
plt.show()
```

CDF_Latencia_all.py

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Leer los datos del archivo iperf y almacena en un data frame
iperf_data = pd.read_csv('C:/Users/javij/OneDrive/Escritorio/Teleco4/2
    Cuatri/TFG/Resultados/JITEL5G/TFG_Javi/Parsed2-Prueba/rtt_parsed_3
    .csv')

tecnologia_vector=['4G', '5GNSA', '5GSA']

df_grouped = iperf_data.groupby('Tecnologia')['RTTavg'].mean().
    reset_index()
df_grouped['RTTavg'] /= 2
# Grafica el promedio del RTTavg para cada tecnologia
plt.bar(df_grouped['Tecnologia'], df_grouped['RTTavg'])
plt.xlabel('RTTavg')
plt.ylabel('RTTavg Promedio')
plt.title('Comparacion de RTTavg por tecnologia')

plt.show()


## CDF depende de MIMO
antenas_vector =['SISO', 'MIMO']
for tecnologia in tecnologia_vector:
    df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]
    for antena in antenas_vector:

        df_filtrado2 = df_filtrado[df_filtrado['MIMO'] == antena]

        df_filtrado2.sort_values(by='RTTavg', inplace=True) #Ordena y
            remplaza en el df_filtrado

        df_filtrado2['Latencia'] = df_filtrado2['RTTavg'].div(2)

        df_filtrado2['CDF'] = np.arange(len(df_filtrado2)) / len(
            df_filtrado2)


        plt.plot(df_filtrado2['RTTavg'], df_filtrado2['CDF'], label=
            tecnologia + " " + antena)
```

```
37
38
39 plt.xlabel('RTTavg')
40 plt.ylabel('CDF')
41 plt.title('CDF latencia dependiendo del numero antenas')
42 plt.grid(True)
43 plt.legend()
44 plt.show()
45
46 ### CDF depende de DUPLEX
47 duplex_vector =['FDD', 'TDD']
48 for tecnologia in tecnologia_vector:
49     df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]
50     for duplex in duplex_vector:
51
52
53         df_filtrado2 = df_filtrado[df_filtrado['Duplex'] == duplex]
54
55         df_filtrado2.sort_values(by='RTTavg', inplace=True) #Ordena y
               remplaza en el df_filtrado
56         df_filtrado2['Latencia'] = df_filtrado2['RTTavg'].div(2)
57
58         df_filtrado2['CDF'] = np.arange(len(df_filtrado2)) / len(
               df_filtrado2)
59
60
61         plt.plot(df_filtrado2['RTTavg'], df_filtrado2['CDF'], label=
               tecnologia + " " + duplex)
62
63
64 plt.xlabel('RTTavg')
65 plt.ylabel('CDF')
66 plt.title('CDF latencia dependiendo de la duplexacion')
67 plt.grid(True)
68 plt.legend()
69 plt.show()
70
71 ## CDF depende de BW
72 BW_vector =[5, 10, 15, 20, 40, 50]
73 for tecnologia in tecnologia_vector:
74     df_filtrado = iperf_data[iperf_data['Tecnologia'] == tecnologia]
75
76     if tecnologia == '5GSA':
77
78         BW_range = BW_vector
79     else:
80
81         BW_range = BW_vector[:4]
82
83     for BW in BW_range:
84
85
86         df_filtrado2 = df_filtrado[df_filtrado['BW'] == BW]
87
88         df_filtrado2.sort_values(by='RTTavg', inplace=True) #Ordena y
               remplaza en el df_filtrado
89
90         df_filtrado2['Latencia'] = df_filtrado2['RTTavg'].div(2)
91
92         df_filtrado2['CDF'] = np.arange(len(df_filtrado2)) / len(
               df_filtrado2)
93
```

```
94
95          plt.plot(df_filtrado2['RTTavg'], df_filtrado2['CDF'], label=
                tecnologia + " " + str(BW)+"MHz")
96
97
98  plt.xlabel('RTTavg')
99  plt.ylabel('CDF')
100 plt.title('CDF latencia dependiendo del ancho de banda')
101 plt.grid(True)
102 plt.legend()
103 plt.show()
```