

UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Integración de redes LoRaWAN y redes 5G

Autor

Juan Guillermo García-Valdecasas Mendoza

Directores

Jorge Navarro Ortiz y Félix Delgado Ferro



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 24 de junio de 2024

Integración de redes LoRaWAN y redes 5G

Autor
Juan Guillermo García-Valdecasas Mendoza

Directores
Jorge Navarro Ortiz y Félix Delgado Ferro

Departamento de Teoría de la Señal, Telemática y Comunicaciones

—

Granada, 24 de junio de 2024

Integración de redes LoRaWAN y redes 5G

Juan Guillermo García-Valdecasas Mendoza

Palabras clave: LoRaWAN, *core* 5G, Cliente NWu, LoRa, N3IWF, ChirpStack, free5gc, Servidores.

Resumen

En la última década, la interconexión entre personas y dispositivos electrónicos ha avanzado notablemente gracias a tecnologías como Internet y las redes móviles inalámbricas. Un componente clave de esta evolución es el *Internet de las Cosas* (IoT), que ha permitido conectar una amplia variedad de dispositivos entre sí y con las personas.

En este contexto, las redes *Long Range Wide Area Network* (LoRaWAN), una tecnología *Low Power Wide Area Network* (LPWAN), han cobrado gran importancia, ya que facilitan la comunicación a grandes distancias con un consumo energético bajo y en intervalos periódicos.

A su vez, la tecnología 5G ha ganado relevancia por su capacidad de proporcionar una conectividad extremadamente rápida y segura a una gran cantidad de dispositivos. Esta característica es fundamental para satisfacer los crecientes requisitos de conectividad masiva en un mundo cada vez más digitalizado.

En este proyecto, se describe una red para la integración de las redes LoRaWAN y 5G, para aumentar la cobertura de la red 5G actual, lo que podría significar mejoras en diversos aspectos del desarrollo de la economía actual. En el proyecto, se explican la implementación y configuración de los elementos que componen la red. Además, también, se incluyen una serie de pruebas para demostrar el correcto funcionamiento de la red implementada.

Integración de redes LoRaWAN y redes 5G

Juan Guillermo García-Valdecasas Mendoza

Keywords: LoRaWAN, *core* 5G, Cliente NWu, LoRa, N3IWF, ChirpStack, free5gc, Servidores.

Abstract

In the last decade, the interconnection between people and electronic devices has advanced notably thanks to technologies such as the Internet and wireless mobile networks. A key component of this evolution is the *Internet of Things* (IoT), which has enabled a wide variety of devices to connect with each other and with people.

In this context, *Long Range Wide Area Network* (LoRaWAN), a *Low Power Wide Area Network* (LPWAN) technology, has gained significant importance as it facilitates communication over long distances with low energy consumption at periodic intervals.

Simultaneously, 5G technology has gained relevance for its ability to provide extremely fast and secure connectivity to a large number of devices. This feature is essential to meet the growing requirements of massive connectivity in an increasingly digitalized world.

In this project, a network for the integration of LoRaWAN and 5G networks is described to increase the coverage of the current 5G network, which could mean improvements in different aspects of the current economic development. The project explains the implementation and configuration of the elements that make up the network. Additionally, a series of tests are included to demonstrate the correct functioning of the implemented network.

Yo, **Juan Guillermo García-Valdecasas Mendoza**, 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 XXXXXXXXX, 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: Juan Guillermo García-Valdecasas Mendoza

Granada a 24 de junio de 2024.

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

D. **Félix Delgado Ferro**, estudiante de doctorado en el Departamento Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Integración de redes LoRaWAN y redes 5G***, ha sido realizado bajo su supervisión por **Juan Guillermo García-Valdecasas Mendoza**, 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 24 de junio de 2024.

El director:

Jorge Navarro Ortiz **Félix Delgado Ferro**

Agradecimientos

Me gustaría comenzar agradeciendo a mis padres por haberme apoyado durante los cuatro años de carrera en todo momento; y, haber confiado en mí y en mis posibilidades. También, me gustaría agradecerles la educación que me han inculcado desde pequeño y por enseñarme que hay que esforzarse para conseguir lo que uno se propone.

También, quiero agradecer a mis amigos que he hecho a lo largo de estos cuatro años y con los que he compartido muy buenos momentos, tanto en clase como fuera de ella. Y, que me han aportado de una forma u otra a mejorar día a día.

Por último, me gustaría agradecer a D. Jorge Navarro Ortiz y a D. Félix Delgado Ferro, por su gran apoyo, disponibilidad y tiempo dedicado a lo largo de todo el proyecto; por haberme guiado en su realización, aportarme conocimiento y ayudarme en todo momento. Gracias a ellos he podido realizar este proyecto.

A todos vosotros, muchas gracias.

Índice general

1. Introducción **27**
1.1. Contexto y motivación 27
1.2. Objetivos y logros alcanzados 29
1.2.1. Objetivos 29
1.2.2. Logros alcanzados 29
1.3. Metodología 30
1.4. Estructura de la memoria 31

2. Estado del arte **35**
2.1. Alternativas para el despliegue de la Red LoRaWAN 35
2.1.1. ChirpStack 35
2.1.2. The Things Network 36
2.1.3. Loriot 37
2.2. Alternativas para el despliegue del *Core* 5G 37
2.2.1. Open5gs 38
2.2.2. Free5gc 39
2.2.3. Open Air Interface 40
2.3. Integración LoRaWAN y 5G 41
2.3.1. Emulador Networking Unit (NWu) 41
2.3.2. UERANSIM 43
2.3.3. srsRAN 44
2.3.4. Proyectos Relevantes 45
2.4. Herramientas Seleccionadas para el Proyecto 47

3. Fundamento teórico **49**
3.1. Long Range Wide Area Network (LoRaWAN) 49
3.1.1. Long Range (LoRa) 49
3.1.2. Capas del protocolo LoRaWAN 51
3.1.3. Arquitectura y seguridad de la red LoRaWAN 52
3.1.4. Clases de dispositivos 55
3.1.5. Message Queing Telemetry Transport 57
3.2. Red 5G 59
3.2.1. Descripción y escenarios en 5G 59

3.2.2. 3GPP 60
3.2.3. Arquitectura de la red 5G 61

4. Planificación y coste **65**
4.1. Planificación temporal 65
4.2. Recursos y costes 67
4.2.1. Recursos software 67
4.2.2. Recursos hardware 69
4.2.3. Recursos humanos 71
4.2.4. Coste total 71

5. Diseño e implementación **73**
5.1. Diseño del modelo de arquitectura para la Integración de LoRaWAN y 5G 73
5.2. Implementación y Configuración de las Motas 74
5.3. Implementación y Configuración del *Gateway* 76
5.4. Implementación y Configuración de la Plataforma LoRaWAN 78
5.5. Implementación y Configuración del *Core* 5G 81
5.6. Implementación y Configuración del cliente *Networking Unit* (NWu) 89

6. Pruebas de conectividad **93**
6.1. Conectividad entre la mota y el *gateway* 93
6.2. Conectividad entre la mota y la Plataforma LoRaWAN . . . 94
6.3. Conectividad entre el cliente NWu y el *Core* 5G 96
6.4. Conectividad entre la mota y la Plataforma LoRaWAN a través del *Core* 5G, sin que participe el cliente NWu 100
6.5. Conectividad End-to-End (E2E) 104

7. Conclusiones y líneas futuras **109**
7.1. Conclusiones 109
7.2. Líneas futuras 110

A. Manual de Usuario **115**

B. Instalación y Configuración de ChirpStack **121**

C. Instalación de free5gc **125**

D. Instalación de NWu-Non3GPP-5GC client **129**

Índice de figuras

1.1. Pila de protocolos de IoT [3] 28
1.2. Diagrama de la metodología del proyecto 31

2.1. Arquitectura de ChirpStack [8] 36
2.2. Arquitectura de The Things Network [10] 37
2.3. Arquitectura de Open5gs [16] 39
2.4. Arquitectura de free5gc [18] 40
2.5. Implementación de RAN en OAI [20] 40
2.6. Implementación del *core* de la red 5G en OAI [21] 41
2.7. Autenticación de un dispositivo Non-3GPP con el 5GC [22] . 42
2.8. Autenticación de un dispositivo Non-3GPP con el 5GC siguiendo el proceso EAP-AKA [22] 42
2.9. Autenticación de un dispositivo Non-3GPP con el 5GC siguiendo el proceso 5G-AKA [22] 43
2.10. Arquitectura de srsRAN [25] 45

3.1. Espectrograma de una transmisión LoRa [31] 50
3.2. Capas del protocolo LoRaWAN [30] 52
3.3. Trama LoRaWAN [30] 52
3.4. Arquitectura de una red LoRaWAN [29] 53
3.5. Encriptado de una trama [30] 54
3.6. Autentificación de una trama [30] 54
3.7. Activación mediante ABP [30] 55
3.8. Activación mediante OTAA [30] 55
3.9. Comunicación de un dispositivo de clase A [30] 56
3.10. Comunicación de un dispositivo de clase B [30] 56
3.11. Comunicación de un dispositivo de clase C [30] 57
3.12. Ejemplo de *4-way handshaking* [37] 59
3.13. Ejemplo de comunicación del protocolo MQTT [35] 59
3.14. Escenarios de uso de 5G [40] 60
3.15. Arquitectura de la red 5G [40] 63
3.16. Arquitectura de la red NSA [43] 63
3.17. Arquitectura de la red SA [43] 64

4.1. Diagrama de Gantt 67
4.2. Mota TTGO LORA32-OLED 70
4.3. Pygate 70

5.1. Diseño de la Arquitectura para la Integración de las redes LoRaWAN y 5G 74
5.2. Configuración motas [55] 75
5.3. Envío del mensaje de una mota visto desde Arduino [55] . . . 75
5.4. Información de la pantalla de la mota [55] 76
5.5. Inicialización del *gateway*, detección de la red Wi-Fi [55] . . . 77
5.6. Inicialización del *gateway*, canales y IP destino [55] 78
5.7. Creación de *gateways* en ChirpStack 79
5.8. Creación de perfil de dispositivo en ChirpStack 79
5.9. Configuración de dispositivo en aplicación de ChirpStack . . . 80
5.10. Configuración de dispositivo en aplicación de ChirpStack (activación) 80
5.11. Comprobación de la Plataforma LoRaWAN en ChirpStack . . 81
5.12. Configuración del AMF en free5gc 82
5.13. Configuración del SMF en free5gc 83
5.14. Configuración del SMF en free5gc 83
5.15. Configuración del UPF en free5gc 84
5.16. Configuración del PCF en free5gc 84
5.17. Configuración del NRF en free5gc 84
5.18. Configuración del UDM en free5gc 85
5.19. Configuración del AUSF en free5gc 85
5.20. Configuración del NSSF en free5gc 85
5.21. Configuración del UDR en free5gc 86
5.22. Configuración del N3IWF en free5gc 87
5.23. Interfaz web de free5gc 87
5.24. Configuración de suscriptor en free5gc 88
5.25. Configuración de reenvío de puertos en la máquina virtual del cliente NWu 89
5.26. Configuración de archivo para arrancar el cliente NWu 90
5.27. Configuración de enrutamiento de paquetes UDP en el cliente NWu 91

6.1. Paquetes de datos recibidos por el *gateway* que han sido enviados por la mota 94
6.2. Paquetes de datos que llegan desde el gateway a la maquina virtual donde están implementados los servidores LoRaWAN 95
6.3. Paquetes de datos recibidos por el *gateway* desde la interfaz web de ChirpStack 95
6.4. Paquetes de datos recibidos por el el servidor LoRaWAN, vistos desde la aplicación creada en ChirpStack 96

6.5. Lanzamiento del servidor del *core* 5G en funcionamiento usando free5gc . . . 97
6.6. Conexión entre el cliente NWu y el *core* 5G (estado 1) 98
6.7. Conexión entre el cliente NWu y el *core* 5G (estado 2) 98
6.8. Conexión entre el cliente NWu y el *core* 5G (estado 3) 99
6.9. Conexión entre el cliente NWu y el *core* 5G (estado 4) 99
6.10. Conexión entre el cliente NWu y el *core* 5G (estado 5) 99
6.11. Conexión entre el cliente NWu y el *core* 5G (estado 6) 100
6.12. Conexión entre el cliente NWu y el *core* 5G (estado 7) 100
6.13. Comprobación con *tcpdump* de que llegan los mensajes de la mota a la máquina del *core* 5G . . . 101
6.14. Comprobación de *iptables* en la máquina virtual donde se implementa el *core* 5G . . . 101
6.15. Comprobación de reenvío de mensajes en *Wireshark* del *core* 5G al servidor LoRaWAN, en la máquina virtual del *core* 5G 101
6.16. Comprobación de reenvío de mensajes en *Wireshark* del *core* 5G al servidor LoRaWAN, en la máquina virtual del servidor LoRaWAN . . . 102
6.17. Comprobación de detección de dispositivos en ChirpStack . . 102
6.18. Comprobación de mensaje recibido por el *gateway* desde ChirpStack . . . 103
6.19. Comprobación de mensaje recibido por el servidor LoRaWAN a través del *core* 5G desde la aplicación de ChirpStack 103
6.20. Comprobación del mensaje de la mota recibido por la máquina virtual donde se implementa el cliente NWu . . . 104
6.21. Comprobación de *iptables* en la máquina virtual donde se implementa el cliente NWu . . . 104
6.22. Comprobación del mensaje recibido por el *namespace* de la máquina del cliente NWu . . . 105
6.23. Comprobación del reenvío del mensaje recibido por el N3IWF hacia el servidor LoRaWAN . . . 105
6.24. Comprobación del reenvío del mensaje recibido por el servidor LoRaWAN . . . 105
6.25. Comprobación de detección de dispositivos en ChirpStack, prueba entre puntos finales . . . 106
6.26. Comprobación de mensaje recibido por el *gateway* desde ChirpStack, prueba entre puntos finales . . . 106
6.27. Comprobación de mensaje recibido por el servidor LoRaWAN desde la aplicación de ChirpStack, prueba entre puntos finales 107

A.1. Creación de *gateways* en ChirpStack . . . 116
A.2. Creación de perfil de dispositivo en ChirpStack . . . 116
A.3. Configuración de dispositivo en aplicación de ChirpStack . . . 117

A.4. Configuración de dispositivo en aplicación de ChirpStack (activación) 117
A.5. Detección de dispositivos en ChirpStack 118
A.6. Mensaje recibido por el *gateway* desde ChirpStack 118
A.7. Mensaje recibido por el servidor LoRaWAN desde la aplicación de ChirpStack 119

B.1. Creación de los contenedores de ChirpStack 122

Índice de tablas

3.1. Especificaciones de Bluetooth 5.0, Wi-Fi y LoRa 51

4.1. Costes de Recursos Software 69
4.2. Especificaciones Técnicas del Portátil 70
4.3. Costes de Recursos Hardware 71
4.4. Coste de Recursos Humanos 71
4.5. Costes Totales 72

Glosario

3G Third Generation Mobile Communication.

3GPP Third Generation Partnership Project.

4G Fourth Generation Mobile Communication.

5G Fifth Generation Mobile Communication.

5G-AKA Fifth Generation Mobile Communication - Authentication and Key Agreement.

5GC 5 Generation Core Network.

ABP Activation By Personalization.

ADR Adaptative Data Rate.

AF Aplication Function.

AMF Access Mobility Management Function.

AppSKey Application Session Key.

AUSF Authentication Server Function.

BW Bandwidth.

CP Control Plane.

CR Code Rate.

CRC Cyclic Redundancy Check.

CSS Chirp Spread Spectrum.

CU Centralized Unit.

DevAddr Device Address.

DevEUI Device EUI.

DN Data Network.

DNS Domain Name Server.

DU Distributed Unit.

E2E End-to-End.

EAP Extensible Authentication Protocol.

EAP-AKA Extensible Authentication Protocol - Authentication and Key Agreement.

eMBB enhance Mobile Broadband.

eNB evolved Node B.

EPC Evolved Packet Core.

ePDG evolved Packet Data Gateway.

EUI64 Extended Unique Identifier 64 bits.

FHSS Frequency Hope Spread Spectrum.

FSK Frequency Shift Keying.

gNB Next-Generation Node B.

GPS Global Positioning System.

GSM Global System for Mobile Communication.

GTP GPRS Tunneling Protocol.

HSS Home Subscriber Server.

IA Inteligencia Artificial.

ID Identification.

IKE Internet Key Exchange.

IKEv2 Internet Key Exchange version 2.

IoT Internet of Things.

IP Internet Protocol.

IPSec Internet Protocol Security.

LoRa Long Range.

LoRaWAN Long Range Wide Area Network.

LPWAN Long Power Wide Area Network.

LTE Long Term Evolution.

MAC Medium Access Control.

MIC Message Integrity Control.

MIMO Multiple-Input Multiple-Output.

MME Mobility Management Entity.

mMTC massive Machine-Type Communications.

MNO Mobile Network Operator.

MQTT Message Queing Telemetry Transport.

N3IWF Non-3GPP Interworking Function.

NAS Non-Access Stratum.

NAT Network Address Translation.

NB-IoT Narrow-Band IoT.

NEF Network Exposure Function.

NF Network Functions.

NGAP Next Generation Application Protocol.

Non-3GPP Non Third Generation Partnership Project.

NR New Radio.

NRF Network Repository Function.

NSA Non Stand Alone.

NSS Network Slice Selection.

NSSF Network Slice Selection Function.

NwkSKey Network Session Key.

NWu Networking Unit.

OAI Open Air Interface.

OTAA Over the Air Activation.

PCF Policy Control Function.

PCRF Policy and Charging Rules Function.

PDCP Packet Data Convergence Protocol.

PFCP Packet Forwarding Control Protocol.

PGWC Packet Gateway Control Plane.

PGWU Packet Gateway User Plane.

QoS Quality of Service.

RAN Radio Access Network.

RLC Radio Link Control.

S-NSSAI Single Network Slice Selection Assistance Information.

SA Stand Alone.

SA Security Association.

SBA Service Based Arquitecture.

SDAP Service Data Adaptation Protocol.

SF Spreading Factor.

SGWC Serving Gateway Control Plane.

SGWU Serving Gateway User Plane.

SMF Session Management Function.

SMS Short Message Service.

SNR Signal to Noise Ratio.

TCP Transmission Control Protocol.

TTN The Things Network.

UDM Unified Data Management.

UDP User Datagram Protocol.

UDR Unified Data Repository.

UE User Equipment.

UP User Plane.

UPF User Plane Function.

URLLC Ultra-Reliable and Low Latency Communications.

USIM Universal Subscriber Identity Module.

Wi-Fi Wireless Fidelity.

WSN Wireless Sensor Networks.

Capítulo 1

Introducción

En este capítulo, se explicará el contexto actual sobre las redes *Long Range Wide Area Network* (LoRaWAN) y la red *Fifth Generation Mobile Communication* (5G); y, la motivación de la integración de estas redes. Así mismo, se introducirán los objetivos y logros alcanzados del proyecto, la metodología para su realización y, finalmente, una descripción de los apartados que lo componen.

1.1. Contexto y motivación

En los últimos años, el mundo de la tecnología relacionado con la comunicación a distancia ha crecido significativamente, facilitando el día a día a las personas. Esto se ha conseguido gracias a avances como Internet y las tecnologías móviles inalámbricas. La evolución ha sido tan grande que hoy en día nos encontramos en un mundo interconectado, tanto entre personas como entre un gran número de dispositivos electrónicos. En esta interconexión, el *Internet de las Cosas* (IoT) desempeña un papel muy importante, permitiendo que nos comuniquemos con dispositivos y que ellos se comuniquen entre sí. Además, IoT permite la recopilación automática de datos, lo que facilita la implementación de mejoras e innovaciones continuas. Esto no solo hace que los dispositivos sean cada vez más eficientes, sino que también mejora nuestra vida diaria de forma segura y rápida.[1]

En el desarrollo de IoT están teniendo una gran importancia las redes LoRaWAN, que utiliza la modulación *Long Range* (LoRa), haciendo que dispositivos se puedan comunicar a grandes distancias con un bajo consumo cada un cierto intervalo de tiempo. Estas redes están jugando un papel muy importante en el desarrollo de las *Smart cities* [2]. LoRaWAN, es una tecnología *Long Power Wide Area Network* (LPWAN) como también podrían ser SigFox, *Narrow-Band IoT* (NB-IoT) o ZigBee. La tecnología IoT se divide en la siguiente pila de protocolos (véase la Fig.1.1): capa de dispositivos,

en la que se encuentran los sistemas físicos como sensores; capa de red, que se divide en la capa de tecnologías LPWAN; la capa *middleware*, que es la capa más importante en la que se encuentra la base de datos y las tecnologías avanzadas como *fog computing*, *edge computing*, *cloud computing* o inteligencia artificial (IA); y la capa de aplicación, en la que se encuentran las aplicaciones con las que interactuamos a través de Internet. [3]

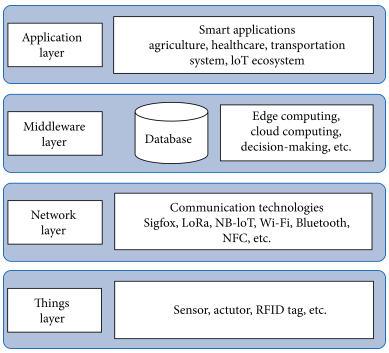


Figura 1.1: Pila de protocolos de IoT [3]

Igualmente, ha sido muy relevante la tecnología 5G, que permite navegar a una velocidad de hasta 10 Gbps y conectar una gran cantidad de dispositivos de forma segura. Esto es fundamental, ya que está aumentando masivamente los requisitos de conectividad entre múltiples dispositivos debido a la evolución de la inteligencia artificial, la evolución de la automatización y el aumento del tráfico de datos móviles debido al crecimento del uso generalizado dispositivos que consumen altos volúmenes de datos móviles, como los *smartphones*. Este motivo, ha impulsado a los investigadores y diseñadores de sistemas a desarrollar diseños de red más eficientes para el futuro. Este desarrollo tiene un gran impacto en las empresas y en la economía actual [4].

Por esto, la principal motivación del proyecto es la integración de las redes LoRaWAN y 5G, ya que podría aumentar la cobertura de la red en zonas rurales. Lo que beneficiaría a la agricultura y a la ganadería, haciendo que evolucionen de forma considerable las granjas inteligentes, ya que se podrían controlar a una mayor distancia y automatizar las que se encuentren en zonas lejanas. Asimismo, tendría un papel importante en la sanidad y la seguridad haciendo que se puedan comunicar emergencias desde donde actualmente no hay cobertura. Además, las redes LoRaWAN no están estandarizadas por *Third Generation Partnership Project* (3GPP) y, por tanto, es interesante esta integración debido a que aportan a las *Mobile Network Operator* MNO funcionalidades y beneficios que las redes 5G estandarizadas no ofrecen. Por otro lado, tendría repercusión económica ya que se integraría

una red que utiliza una tecnología con un precio más asequible y con una larga duración de baterías y consumo, LoRaWAN; con una red con baja latencia y alta tasa de transmisión de datos, 5G. Esta integración podría dar lugar a que las compañías móviles puedan usar ambas tecnologías y aumente su cobertura, impulsando así la innovación y la competencia en el mercado de las telecomunicaciones [5] [6].

1.2. Objetivos y logros alcanzados

En este apartado, se definen los principales objetivos y logros alcanzados en este proyecto.

1.2.1. Objetivos

Los principales objetivos que se quieren lograr en este proyecto son:

1. Análisis y estudio de la red LoRaWAN, comprendiendo su funcionamiento, características y aplicaciones en entornos similares al del proyecto.
2. Análisis y estudio de la red 5G, comprendiendo su funcionamiento, características y aplicaciones en entornos similares al del proyecto.
3. Diseño de un modelo de arquitectura que permita la implementación de la integración de las redes LoRaWAN y 5G.
4. Implementación del modelo de arquitectura diseñado para la interoperabilidad entre dispositivos LoRaWAN y 5G.
5. Pruebas y evaluación de la arquitectura implementada para la integración de las redes LoRaWAN y 5G.

1.2.2. Logros alcanzados

Los logros alcanzados durante la realización del proyecto son:

1. Recopilación de información sobre la tecnología LoRaWAN, comprendiendo su funcionamiento y principales aplicaciones en la actualidad.
2. Recopilación de información sobre la tecnología 5G, comprendiendo su funcionamiento y principales aplicaciones en la actualidad.
3. Recopilación de información sobre artículos recientes con un entorno similar al de este proyecto.
4. Diseño e implementación de un modelo de arquitectura que permite la integración de las redes LoRaWAN y 5G.

5. Comprobación del correcto funcionamiento del modelo de arquitectura disñado e implementado.

1.3. Metodología

En esta sección, se va a exponer la metodología empleada para la realización del proyecto y comprobación de éxito de los objetivos planteados en el apartado 1.2. Se va a seguir una línea de trabajo secuencial, de forma que se puede comprobar el avance del proyecto de forma clara permitiendo la comprobación y adaptación de cada etapa, en caso de que fuese necesario.

En primer lugar, se define la estructura del proyecto de forma ordenada y clara; y los objetivos que se pretenden alcanzar con su realización. A su vez, se lleva a cabo la temporarización del trabajo. Después, se analizan distintas soluciones para el diseño de la arquitectura para la integración de las redes LoRaWAN y 5G, teniendo en cuenta las herramientas que hay en la actualidad y sus características. Una vez elegida la solución, se procede con la recopilación de información sobre estudios realizados recientemente relacionados con el contexto del proyecto. Seguido de esto, se recopila información teórica sobre las redes LoRaWAN y 5G.

El siguiente paso, es realizar el diseño atendiendo a cuál es el funcionamiento de la red que se quiere lograr. Acto seguido, se continúa con la implementación de la red LoRaWAN y del core 5G por separado, realizando la correcta instalación de las herramientas software que se van a utilizar y una preparación de los entornos necesarios para su funcionamiento. Al mismo tiempo, se realiza la programación de las motas y pasarelas. Posteriormente, se procede a hacer la integración de ambas redes.

Finalmente, se lleva a cabo la realización de distintas pruebas de conectividad punto a punto y *End-to-End* (E2E) para la correcta comprobación del funcionamiento de la red para concluir si se han logrado los objetivos propuestos.

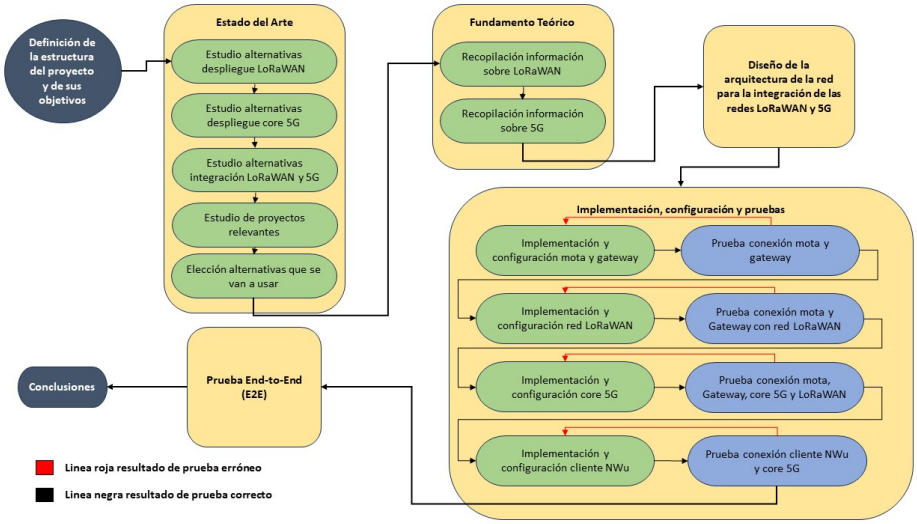


Figura 1.2: Diagrama de la metodología del proyecto

1.4. Estructura de la memoria

En esta sección, se van a exponer los diferentes capítulos que conforman la memoria del proyecto, explicando brevemente de que trata cada uno:

- **Capítulo 1. Introducción**: Este capítulo esta compuesto por las siguientes partes:
  - *Contexto y motivación*: expone el ámbito del proyecto y la problemática que se quiere resolver con él y sus beneficios.
  - *Objetivos y logros alcanzados*: expone los principales objetivos del proyecto y cuáles se han logrado.
  - *Metodología*: describe los pasos seguidos durante la realización del proyecto.
  - *Estructura de la memoria*: describe los capítulos que componen la memoria y su contenido.
- **Capítulo 2. Estado del arte**:
  - *Alternativas para el despliegue de la Red LoRaWAN*: presenta las diferentes herramientas y software que se pueden emplear para el despliegue de la red LoRaWAN.
  - *Alternativas para el despliegue del Core 5G*: presenta las diferentes herramientas que se pueden emplear para el despliegue de la red 5G.

  - *Integración LoRaWAN y 5G*: se tratan distintas tecnologías para la integración de las redes LoRaWAN y 5G. Además, se presentan distintos estudios realizados recientemente en un contexto similar al de este proyecto.
  - *Herramientas Seleccionadas para el Proyecto*: se explican cuáles son las herramientas seleccionadas para la realización del proyecto y su motivo.

- **Capítulo 3. Fundamento teórico**: Este capítulo explica los conceptos teóricos necesarios para la compresión y realización del proyecto. Se divide en:

  - *LoRaWAN*: explicación de las características principales de esta tecnología.
  - *Red 5G*: explicación de las características de esta red, principalmente su arquitectura.

- **Capítulo 4. Planificación y coste**: Este capítulo se divide en los siguientes apartados:

  - *Planificación temporal*: muestra la organización temporal durante la realización del proyecto.
  - *Recursos y costes*: trata los distintos recursos hardware, software y humanos utilizados; y una estimación del coste de cada uno de ellos y en total.

- **Capítulo 5. Diseño e implementación**: Este capítulo describe el diseño y la implementación del proyecto. Se subdivide en los siguientes apartados:

  - *Diseño del modelo de arrquitectura para la Integración de LoRaWAN y 5G*: explica el diseño de la red para la realización del proyecto y los elementos que la componen.
  - *Implementación y Configuración de las Motas*: muestra como se configuran las motas *TTGO LORA32-OLED* usadas en el proyecto.
  - *Implementación y Configuración del Gateway*: muestra como se configura el *gateway Pygate* usado en el proyecto.
  - *Implementación y Configuración de la Plataforma LoRaWAN*: muestra como se configuran los servidores LoRaWAN de la red usando ChirpStack.
  - *Implementación y Configuración del Core 5G*: muestra como se configura el servidor del *core* 5G y los elementos que lo componen usando free5gc.

  - *Implementación y Configuración del cliente Networking Unit (NWu)*: muestra como se configura el cliente NWu para conectarse con el *Non-3GPP Interworking Function* (N3IWF).

- **Capítulo 6. Pruebas de conectividad**: Este capítulo expone las pruebas realizadas y los resultados obtenidos de ellas. Se subdivide en las siguientes pruebas:

  - *Conectividad entre la mota y el gateway*: se comprueba que los mensajes enviados por la mota llegan correctamente al *gateway*.
  - *Conectividad entre la mota y los servidores LoRaWAN*: se comprueba que los mensajes enviados por la mota llegan correctamente a los servidores LoRaWAN a través del *gateway*.
  - *Conectividad entre el cliente NWu y el core 5G*: se comprueba la conexión y autenticación entre el cliente NWu y el *core* 5G.
  - *Conectividad entre la mota y la Plataforma LoRaWAN a través del core 5G, sin que participe el cliente NWu*: se comprueba que los mensajes enviados por la mota llegan correctamente al servidor LoRaWAN a través del *core* 5G.
  - *Conectividad End-to-End (E2E*: se comprueba que los mensajes enviados por la mota llegan correctamente al servidor LoRaWAN a través de toda la red.

- **Capítulo 7. Conclusiones y líneas futuras**: Este capítulo se subdivide en:

  - *Conclusiones*: trata las conclusuiones obtenidas tras la realización del proyecto.
  - *Líneas futuras*: se tratan posibles líneas que se pueden seguir en el futuro para mejorar su funcionamiento.

- **Apéndice A. Manual de usuario**: Este apartado explica cómo utilizar el proyecto desarrollado para posibles investigaciones, pruebas y mejoras.

- **Apéndice B. Instalación y Configuración de ChirpStack**: Este apartado explica cómo se instala y configura la aplicación ChirpStack.

- **Apéndice C. Instalación de free5gc**: Este apartado explica cómo se instala la herramienta free5gc.

- **Apéndice D.Instalación de NWu-Non3GPP client**: Este apartado explica cómo se instala el cliente NWu.

Capítulo 2

Estado del arte

Este capítulo se divide en cuatro secciones. En la primera sección, se tratan las distintas alternativas para el despliegue de la red LoRaWAN; en la segunda sección, las alternativas del despliegue de la red 5G; en la tercera, la integración de éstas dos redes; y, en la última, se explica cuáles son las herramientas elegidas para realizar el proyecto y su motivo.

2.1. Alternativas para el despliegue de la Red LoRaWAN

A continuación, se van a presentar distintas tecnologías para el despliegue de la red LoRaWAN de código abierto.

2.1.1. ChirpStack

ChirpStack es un servidor de red y aplicación LoRaWAN de código abierto que permite configurar redes de este tipo. Proporciona una interfaz web para la gestión de la red desde la que se pueden configurar los *gateways*, los dispositivos que forman parte de la red y visualizar los paquetes de datos que se transportan en la red, tanto los que llegan a los *gateways* como a los servidores [7].

Soporta dispositivos LoRa de clase A, B y C. La arquitectura se muestra en la figura 2.1. En ella, se puede observar que consta de un *ChirpStack Gateway Bridge* que se conecta a los gateways LoRa a través del puerto *User Datagram Protocol* (UDP) con puerto 1700, y que este envía los paquetes del protocolo UDP al resto del servidor. También se puede observar que tiene un *broker* MQTT en el puerto 1883, lo que permite usar el protocolo MQTT para el envío de los mensajes en el servidor [8].

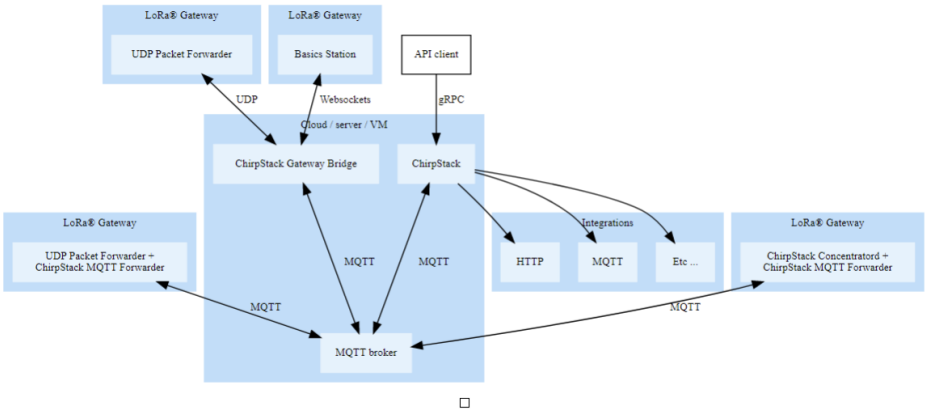


Figura 2.1: Arquitectura de ChirpStack [8]

2.1.2. The Things Network

The Things Network (TTN) proporciona un conjunto de herramientas de código abierto y una red global abierta para construir aplicaciones IoT con bajo coste, de forma segura y escalable [9]. Esta red global es una red LoRaWAN que está compuesta por múltiples dispositivos de usuario final y *gateways*, estos dispositivos utilizan las modulación LoRa y pueden ser de clase A, B y C [10].

TTN permite a los usuarios crear sus propias redes haciendo uso de estos dispositivos e incorporarse a la red global, que cuenta con un servidor de red y de aplicación, autenticándose a través de su *Join Server*. Las entidades que forman parte de la red quedan registradas en el *Identity Server*. La red también incluye un *broker* MQTT que realiza una función similar al de ChirpStack. En la figura 2.2 se muestra la arquitectura de su red [10].

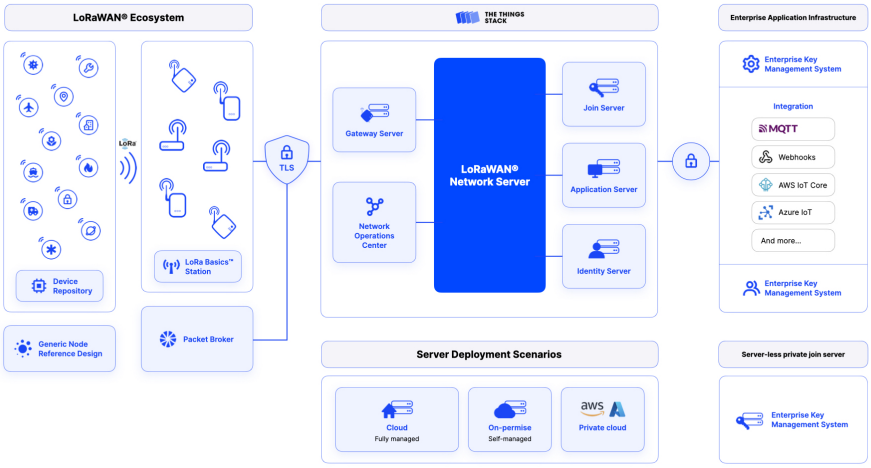


Figura 2.2: Arquitectura de The Things Network [10]

2.1.3. Loriot

Loriot es una plataforma que proporciona soluciones para la conexión masiva de dispositivos IoT [11]. Proporciona servidores de red que utilizan tecnologías LoRaWAN o *mioty*, una tecnología LPWAN diseñada para IoT [12]. Esta plataforma ofrece un servidor gratuito y otro destinado para ámbitos más profesionales de pago, su pago varía mensual o anualmente en función de los servicios que presta.

La arquitecturas con un servidor de red LoRaWAN o *mioty* son similares. Utilizan los dispositivos finales y *gateways*, en el caso de LoRaWAN, o estaciones base, en el caso de *mioty*; que utilizan un protocolo u otro en función del servidor de red. Considerando que se utiliza el servidor LoRaWAN, este se encarga de manejar toda la infraestructura, de la seguridad de la red, el encaminamiento de los mensajes y la autentificación de los dispositivos [13]. Y, en lo que respecta a utilizar el servidor de red *mioty*, este proporciona el manejo de la red, la recolección de datos de los dispositivos de la red, el procesamiento de los datos, la seguridad, la autentificación de los dispositivos y la integración con otros sistemas y plataformas [14]. Ambas opciones de red tienen un servidor de aplicación que permiten analizar los datos de la red. Ambas, aportan una interfaz web para gestionar y modificar la red.

2.2. Alternativas para el despliegue del *Core* 5G

En esta sección, se van a presentar alternativas actuales para implementar la red 5G.

2.2.1. Open5gs

Open5gs es un proyecto de código abierto escrito en lenguaje de programación C que permite implementar un *Fifth Generation Core Network* (5GC) y un *Evolved Packet Core* (EPC) para crear una red privada usando un *Next-Generation Node B* (gNB) o un *evolved Node B* (eNB) y una *Universal Subscriber Identity Module* (USIM) [15]. Permite implementar la red con una arquitectura *Non-Stand Alone* (NSA), es decir, utilizando una red de acceso 5G y en *core* 4G; o una arquitectura *Stand Alone* (SA), utilizando una red de acceso y *core* 5G [16].

La arquitectura de la red NSA tiene el *core* dividido plano de control o *Control Plane* (CP) y en plano de usuario o *User Plane* (UP). El elemento principal del CP es el *Mobility Management Identity* (MME) que controla las seiones, la movilidad y la paginación. Se conecta con el *Home Subscriber Server* (HSS) que se encarga de generar la autentificación y mantener el perfil de los suscriptores; también se conecta con el *Serving Gateway Control Plane* (SGWC) y con *Packet Gateway Control Plane/Session Management Function* (PGWC/SMF), que son el plano de control de los servidores de los *gateways*. Al MME se conectan los eNBs de la red. El último elemento del CP es el *Policy and Charging Rules Function* (PCRF), que se encuentra entre el (PGWC/SMF) y el HSS, se encarga de que los suscriptores cumplan las políticas de la red y de sus cargos. Por otro lado, el UP se encarga de llevar los paquetes de datos desde el eNB/gNB hasta la red externa a través de la red. Está compuesto por el *Serving Gateway User Plane* (SGWU) y el *Packet Gateway User Plane/User Plane Function* (PGWU/UPF), que se conectas a sus respectivos elementos del plano de control. El eNB/gNB se conecta al SGWU, que se conecta con PGWU/UPF, y este con la red externa.

Por otra parte, la arquitectura de la red SA usa en el *core* una *Service Based Architecture* (SBA), y también está dividida en el CP y UP. Las funciones del CP están configuradas para que se registren con el *Network Repository Function* (NRF) y desde este elemento descubran el resto del *core*. El *core* está compuesto por el *Access and Mobility Management Function* (AMF), que controla la conexión y la movilidad, y al que se conectan los eNBs. El *Unified Data Management* (UDM), *Authentication Server Function* (AUSF), y el *Unified Data Repository* (UDR) se encargan de realizar funciones similares al HSS, es decir, generar la autentificación y mantener los perfiles de los suscriptores. El *Session Management Function* (SMF) se encarga de la gestión de las sesiones, el *Network Slice Selection Function* (NSSF) de la selección del *network slice*; y, el *Policy and Charging Function* (PCF) de que se cumplan las políticas de la red y de los cargos a los suscriptores. En el plano de usuario, solo se encuentra el *User Plane Function*

(UPF), que se encarga de llevar los paquetes de datos desde el gNB hasta la red externa y se conecta con el SMF.

En la figura 2.3 se muestra la arquitectura completa de Open5gs:

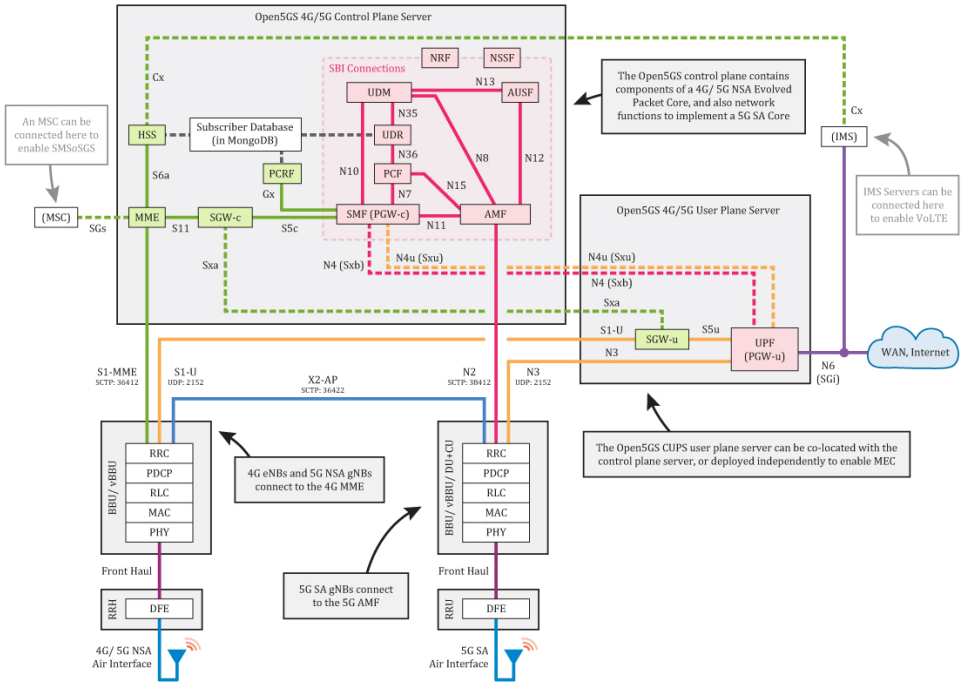


Figura 2.3: Arquitectura de Open5gs [16]

2.2.2. Free5gc

Free5GC también es un proyecto de código abierto cuyo objetivo es principal es implementar el core de la red 5G definida en 3GPP Release 15 (R15) y que proporciona una interfaz web [17].

Este código se puede descargar desde el repositorio [18], en el que se encuentran las implementaciones de los distintos componentes del 5GC: AMF, AUSF, *Non-3GPP Interworking Function* (N3IWF) (lo que permite que se puedan conectar dispositivos Non-3GPP), NRF, NSSF, PCF, SMF, UDM, UDR y UPF. En la figura 2.4 se muestra su arquitectura.

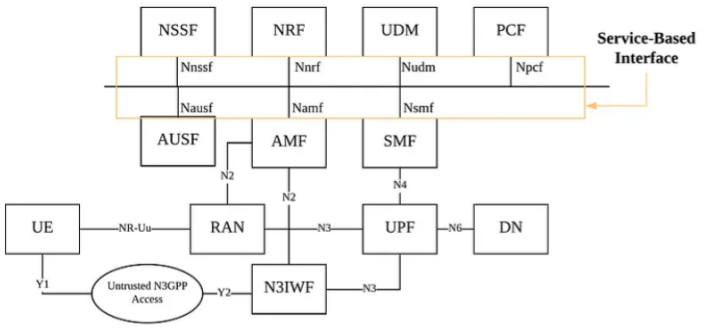


Figura 2.4: Arquitectura de free5gc [18]

2.2.3. Open Air Interface

Open Air Interface (OAI) es una plataforma de código abierto basada en 3GPP que está dividida en dos proyectos, el primero para ofrecer *Radio Access Network* (RAN) tanto de 4G como de 5G; y , el segundo, para proporcionar el *core* de una red 4G o 5G [19].

Permite implementar una red 5G NSA o SA. En la RAN proporciona los nodos eNB y gNB y los dispositivos de usuario, y permite operar con la tecnología *Multiple-Input Multiple-Output* (MIMO).

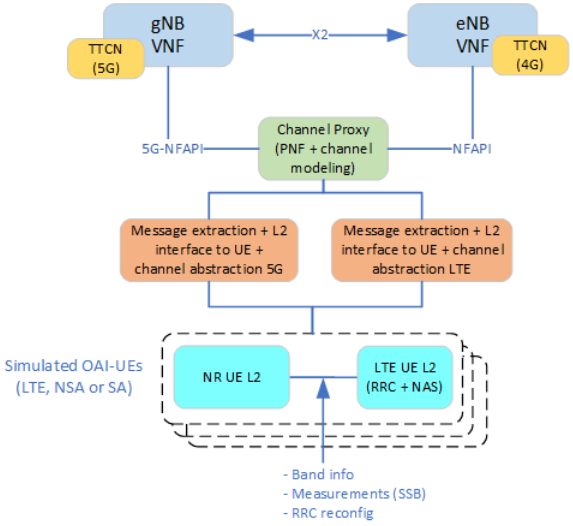


Figura 2.5: Implementación de RAN en OAI [20]

Actualmente, permite implementar por completo una red con una arquitectura NSA. Mientras que la arquitectura SA está en desarrollo y cuenta con los siguientes elementos a día de hoy: SMF, AMF, UPF, NRF, AUSF,

UDM, UDR, PCF y NSSF; y soporta procesos para la conexión, el registro de dispositivos y el control de las sesiones. El desarrollo de este proyecto está siendo probado por profesionales [21].

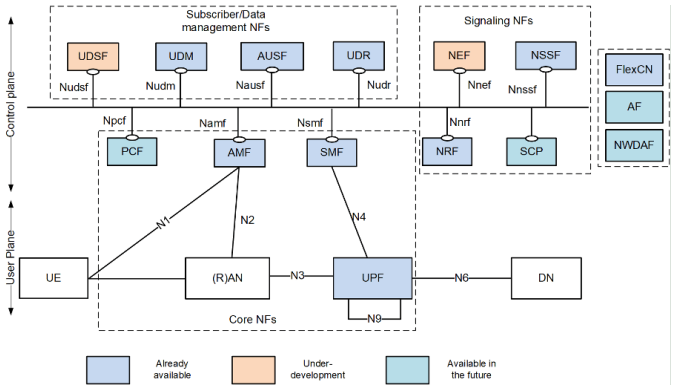


Figura 2.6: Implementación del *core* de la red 5G en OAI [21]

2.3. Integración LoRaWAN y 5G

En esta sección, se van a presentar herramientas que pueden ser utilizadas para la integración de las redes LoRaWAN y 5G.

2.3.1. Emulador Networking Unit (NWu)

El emulador *Networking Unit* (NWu) permite simular un dispositivo en la red sin necesidad de contar con él fisicamente, esto se puede usar para conectar el UE con el 5GC de forma segura a través de él. El emulador NWu, se puede implementar con el repositorio [22]. Es un emulador de cliente NWu realizado en python3 que establece túneles IKEv2/IPSec con N3IWF o *evolved Packet Data Gateway* (ePDG).

El plano de control IKEv2 se utiliza para ayudar a realizar la autenticación y establecer *SA Child*, tanto el de señalización como el de plano de usuario.

El emulador del repositorio puede utilizar cualquier tipo de red(Wi-Fi, fija, móvil o LoRaWAN) para establecer el túnel IPSec hasta el N3IWF o ePDG. Genera cada clave y parámetro utilizados en los procesos IKEv2/IPSec, lo que hace que se puedan decodificar, y, evaluar en *Wireshark*. Además, permite dos métodos de autenticación: EAP-AKA o 5G-AKA.

En la figuras 2.7, se muestra la autenticación de un dispositivo Non-3GPP con el 5GC, y en las figuras 2.8 y 2.9 se muestran los distintos tipos de autenticaciones que permiten (EAP-AKA o 5G-AKA):

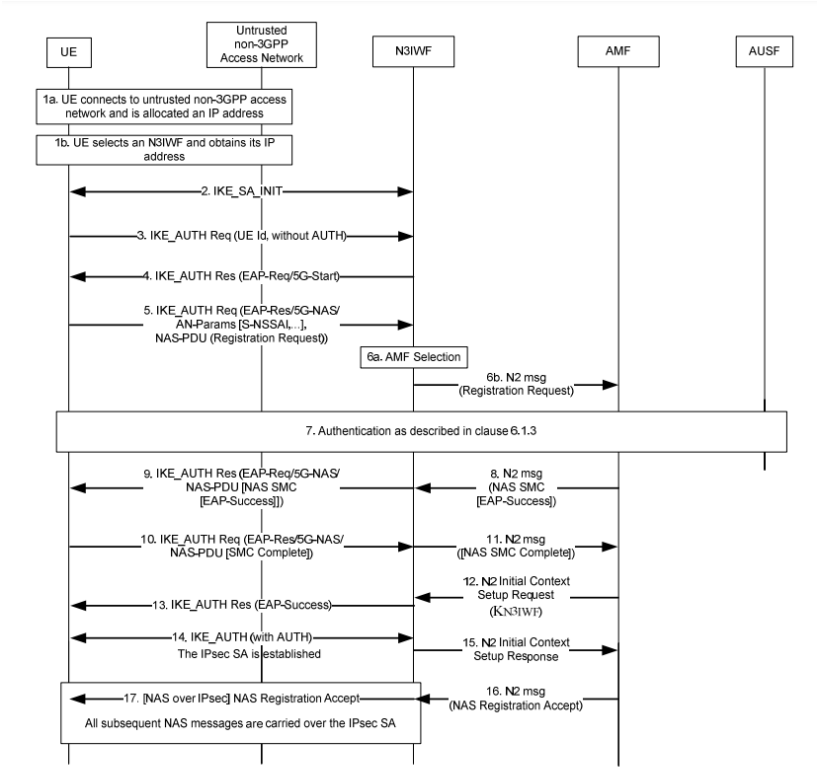
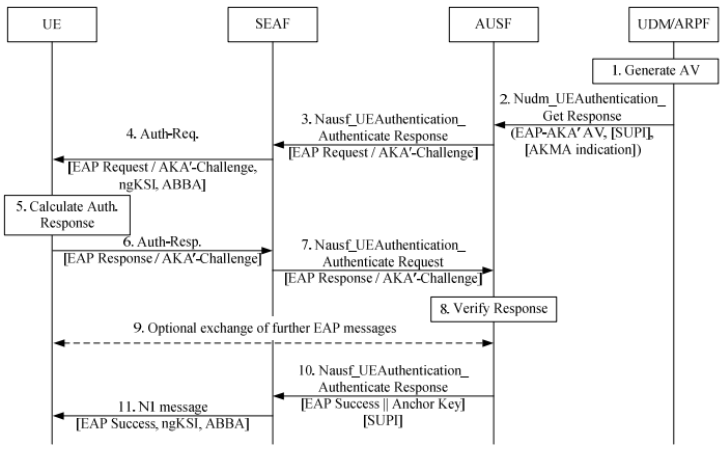


Figure 7.2.1-1: Authentication for untrusted non-3GPP access

Figura 2.7: Autenticación de un dispositivo Non-3GPP con el 5GC [22]



6.1.3.1-1: Authentication procedure for EAP-AKA'

Figura 2.8: Autenticación de un dispositivo Non-3GPP con el 5GC siguiendo el proceso EAP-AKA [22]

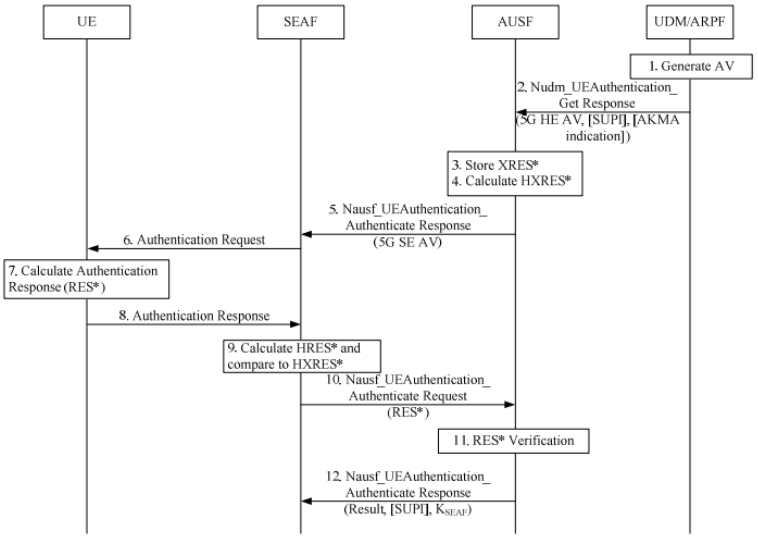


Figure 6.1.3.2-1: Authentication procedure for 5G AKA

Figura 2.9: Autenticación de un dispositivo Non-3GPP con el 5GC siguiendo el proceso 5G-AKA [22]

El emulador tiene 4 procesos distintos:

- El proceso principal en el que se hace el flujo de IKEv2.
- Codificador IPSec: se obtiene los datos del usuario de la interfaz del túnel, los encripta y los manda al N3IWF o ePDG. Y, recibe información del proceso principal para crear, actualizar o eliminar el *SA Child*.
- Decodificador IPSec: recibe los datos encriptados del N3IWF, los desencripta y los manda a la interfaz del túnel. También, le manda el mensaje desencriptado IKEv2 al proceso principal.
- Proceso TCP: intercambia mensajes *Non-Access Stratum* (NAS) con el N3IWF a través de la señalización *SA Child*; y, también intercambia estos mensajes con el proceso principal.

2.3.2. UERANSIM

UERANSIM es un proyecto de código abierto escrito en el lenguaje C++ que permite simular dispositivos de usuario y nodos gNB basándose en 3GPP [23]. Puede ser utilizado para realizar pruebas en redes 5G.

Proporciona interfaces radio simuladas sobre el protocolo UDP. Su arquitectura está dividida en el plano de control y plano de usuario. Haciendo uso del protocolo *Next Generation Application Protocol* (NGAP) en el plano de control, y del protocolo *GPRS Tunneling Protocol* (GTP) en el plano de usuario.

2.3.3. srsRAN

srsRAN también es un software de código abierto que proporciona radio 5G y 4G, desarrollado por Software Radio Systems [24]. Este proyecto, también, sigue los estándares de 3GPP. Su arquitectura se divide en una *distributed unit* (DU), que se divide en una parte superior y otra inferior; y una *centralized unit* (CU), y esta última, a su vez, se divide en CP y UP [24].

La parte superior del DU se encarga de procesar el *Medium Access Control* (MAC) y el *Radio Link Control* (RLC) del tráfico de *uplink* y *downlink*. La parte inferior, se encarga de procesar la parte física de ambos tráficos. Las dos partes, se comunican con el CP y UP

En la *centralized unit*, el CP se encarga del control de las sesiones en la red, especialmente el *Packet Data Convergence Protocol* (PDCP). El UP, se encarga del plano de usuario de las sesiones, tienen gran importancia el PDCP y el *Service Data Adaptation Protocol* (SDAP). En la figura 2.10 se muestra la arquitectura de srsRAN.

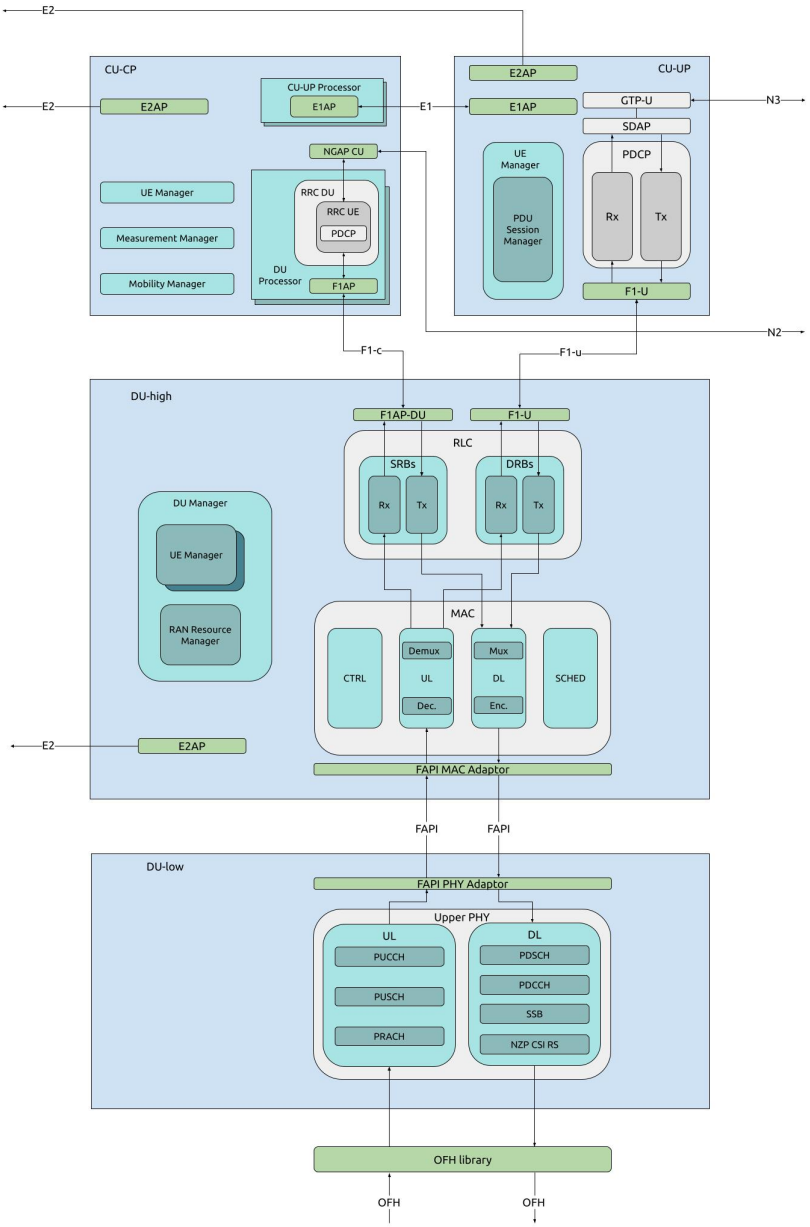


Figura 2.10: Arquitectura de srsRAN [25]

2.3.4. Proyectos Relevantes

El objetivo de esta sección, es presentar proyectos realizados recientemente que muestran la relevancia de la integración de redes LoRaWAN y 5G:

- ***Integration of LoRaWAN and 4G/5G for the Industrial In-***

ternet of Things **[26]**: este proyecto fue publicado en 2018, su objetivo era integrar LoRaWAN con la red móvil 4G/5G para que aplicaciones IoT industriales se integren en esta red. Para conseguir esto, se propuso modificar el *gateway* de LoRaWAN sin modificar su servidor y dispositivos finales, de forma que mantienen la seguridad de LoRaWAN. El *gateway* actúa como un dispositivo final y un ENB para comunicarse con el EPC de la red, y los paquetes de datos se envían a través de este de forma segura y encriptados.

- ***A LoRaWAN Architecture for Communications in Areas without Coverage: Design and Pilot Trials*** **[27]**: este proyecto fue publicado en 2022, su objetivo es implementar un red LoRaWAN para zonas donde no hay cobertura móvil. Para lograr esto, se propusieron dos alternativas: en la primera alternativa, proporciona libertad de movimiento en ambos extremos de la red, es decir, los dispositivos finales y el nodo central podían moverse aunque la arquitectura central estaba centralizada; en la segunda alternativa, la estructura estaba centralizada, de modo que el nodo central se ubicaba en el mismo sitio todo el rato aunque los dispositivos finales si se movían libremente. En el caso de la primera alternativa, los dispositivos finales se conectaban a una mota, y esta mota a un *gateway* que actuaba como puenteLoRa y envía la información al nodo central; por otro lado, en el caso de la segunda, los dispositivos finales se conectaban a una mota, y esta directamente al nodo central. Ambas alternativas fueron probadas en ambientes rurales, naturales y rurales, teniendo buen resultado, siendo los óptimos en las zonas rurales.

- ***A Survey of 5G Emerging Wireless Technologies Featuring LoRaWAN, Sigfox, NB-IoT and LTE-M*** **[28]**: este estudio fue publicado en 2019, su objetivo es examinar distintas tecnologías que pueden ser aplicadas en la comunicación 5G y cómo se pueden utilizar en *Wireless Sensor Networks* (WSN); en concreto, LoRaWAN, Sigfox, NB-IoT and LTE-M. Tras estudiar las distintas tecnologías, concluyeron, que LoRaWAN es una buena solución para WSN con baja tasa de datos, una gran cantidad de nodos por celda y bajo consumo; NB-IoT es buena cuando se necesita una alta tasa de datos, con pocos nodos y sin límites de consumo; Sigfox para cuando la banda del espectro de frecuencias es pequeña; y, LTE-M cuando se necesita una velocidad alta y baja latencia, pero tiene el inconveniente de que tiene un alto consumo.

2.4. Herramientas Seleccionadas para el Proyecto

Tras exponer distintas alternativas para realizar el proyecto, en esta sección, se va a explicar cuáles son las seleccionadas para el despliegue de la red LoRaWAN, el despliegue de la red 5G y la integración de ambas redes.

Para el despliegue de las red LoRaWAN, se ha seleccionado la herramienta ChirpStack en vez de TTN o Loriot. Esta selección se ha hecho por que ChirpStack ofrece un mayor control sobre la red, permitiendo usar dispositivos propios en vez de los que proporciona la herramienta y una mayor cantidad de dispositivos; y, no teniendo que acogerse a las políticas de los proveedores de las redes de TTN y Loriot.

Para el despliegue de la red 5G, se ha optado por free5gc, ya que ofrece N3IWF, permitiendo conexiones fuera de los estándares 3GPP, como es el caso de LoRaWAN; y está completamente desarrollada, al contrario que OAI.

Por último, para la integración de ambas redes, se va a utilizar el emulador NWu, ya que puede ser utilizado para proporcionar una conexión segura con el 5GC, estableciendo un túnel con el N3IWF para cualquier tipo de red mediante IKEv2/IPSec; y, permite los métodos de autenticación EAP-AKA o 5G-AKA. Además, en el repositorio con el que se puede implementar, explica como hacerlo con free5gc, por lo que facilita el desarrollo del proyecto.

Capítulo 3

Fundamento teórico

En este capítulo, se explicarán los fundamentos teóricos de LoRaWAN, explicando la tecnología LoRa y el protocolo *Message Queing Telemetry Transport* (MQTT); también se mostrará una visión general sobre la tecnología 5G.

3.1. Long Range Wide Area Network (LoRaWAN)

LoRaWAN define el protocolo de comunicación y la arquitectura de una tecnología que permite la comunicación a larga distancia y de bajo consumo, siendo LoRa su modulación, aunque se pueden emplear otras modulaciones como *Frequency Shift Keying* (FSK). Esta modulación, afecta al tiempo de duración de la batería de los nodos, la capacidad de los enlaces, la seguridad, la calidad y las aplicaciones que sirve [29].

3.1.1. Long Range (LoRa)

LoRa es la capa física o la modulación inalámbrica utilizada para crear enlaces de comunicación a larga distancia y robusta frente a las interferencias. Utiliza una modulación de espectro ensanchado, *Chirp Spread Spectrum* (CSS), que proporciona baja potencia y un largo alcance. Esta modulación envía pulsos de *Chirp* que aumentan o disminuyen la frecuencia con el tiempo de forma aleatoria [30]. En la figura 3.1 se muestra el espectograma de una transmisión LoRa usando CSS, en la que la frecuencuencia aumenta cuando las lineas del espectograma están en sentido ascendente y disminuyen en el caso contrario[31]:

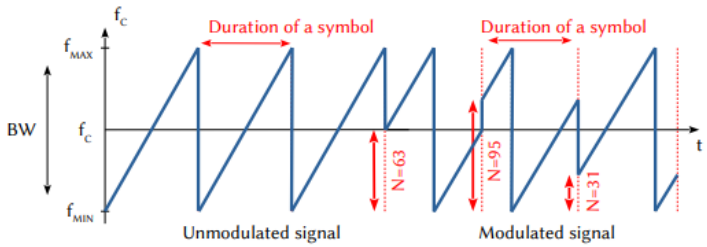


Figura 3.1: Espectrograma de una transmisión LoRa [31]

Utiliza 8 *Spreading Factors* (SF), del 5-12. Según el factor de dispersión que se use, el *Chirp* transportará más bits o menos. Por ejemplo, si se usa el SF10 el Chirp transportará 10 bits, por lo que cuanto mayor es el SF mayor es el tiempo que tarda en trasmitirse un símbolo. La banda de frecuencia en la que trabaja puede ser la 868 MHz con un ancho de banda de 125-500 KHz, aunque también se pueden utilizar otras bandas como la de 433 MHz, 915 MHz, e incluso 2.4 GHz. La tasa de bits teórica (Tb_t) depende del ancho de banda o *Bandwidth* (BW) y del factor de dispersión y de los símbolos por segundo, *Code Rate* (CR).Esta tasa de bits viene expresada ecuación 3.1 [30]:

$$Tb_t = SF \times \left(\frac{BW}{2^{SF}}\right) \times \left(\frac{4}{4+CR}\right) \tag{3.1}$$

Y el tiempo de transmisión de un símbolo ($T_{simbolo}$) depende del ancho de banda y del SF, viene expresado en la ecuación 3.2:

$$T_s = \frac{2^{SF}}{BW} \tag{3.2}$$

La gran ventaja de LoRa es su capacidad de transmitir a grandes distancias, una pasarela o *gateway* puede llegar a cubrir decenas de kilómentros cuadrados. La distancia a la que puede transmitir puede variar según los entornos en los que se encuentren las pasarelas. El balance del enlace (B_{enlace}) es el factor principal para determinar la distancia (D) a la que puede llegar en un entorno, este se expresa en dB. La distancia se puede determinar como en la ecuación 3.3:[29] [30]

$$D = \sqrt{\frac{10^{\left(\frac{B_{enlace}}{10}\right)}}{1755 \times f^2}} \tag{3.3}$$

Una de las principales características de LoRa es el largo tiempo de vida de su batería. Utiliza una batería que está muy optimizada y que consume

muy poca potencia en comparación con otras tecnologías. Esto se debe a que en las redes LoRa, los nodos solo se comunican cuando tienen información que enviar. Es decir, utilizan el método ALOHA, en el que se envían mensajes solo cuando hay datos listos para enviar sin comprobar si hay otras transmisiones en el medio, lo que puede provocar colisiones entre paquetes de datos. Esto lo hacen formando una red en estrella, en la que en medio se encuentra el *gateway*. [32]

La capacidad de transmisión de la red se ve beneficiada por el uso de *Adaptative Data Rate* (ADR) en los nodos y del transceptor multimodem multicanal en el *gateway*. Los nodos pueden activar o desactivar el uso de ADR poniendo a 1 o a 0 el bit de ADR en una transmisión ascendente. Cuando este se activa, se recopila información del *Signal to Noise Ratio* (SNR), de la tasa de datos, del número de *gateways* que reciben datos y de la potencia de la señal de las últimas 20 transmisiones, con esto se incrementa o disminuye la tasa de datos y potencia para optimizar la capacidad de transmisión. El transceptor multimódem multicanal permite recibir mensajes simultáneamente desde múltiples pasarelas. [32]

Una desventaja de esta tecnología es que solo se puede usar en transmisiones que requieren una tasa de transmisión baja; también que al usar ciclos de trabajo limita el número de mensajes que se pueden enviar a la vez [32]. A continuación, se muestra una comparación de la tecnología LoRa con la tecnología Bluetooth y *Wireless Fidelity* (Wi-Fi) [33]:

Especificaciones	**Bluetooth 5.0**	**Wi-Fi**	**LoRa**
Modulación	*Frequency Hope Spread Spectrum* (FHSS)	FHSS	CSS
Frecuencia	2.4 GHz	2.4 y 5 GHz	EU (433, 868 MHz), US (915 MHz), Asia (430 MHz)
Rango	10–100 m	100 m	10 km (aprox.)
Tasa de datos	1, 2, 3 Mbps	0.1–54 Mbps	<50 Mbps
Consumo de energía	Medio	Alto	Bajo
Aplicación	Conectar portátiles, PDAs, teléfonos y auriculares	Conectividad LAN inalámbrica, acceso a Internet de banda ancha	Conectar portátiles, PDAs, teléfonos, auriculares, sensores y otros dispositivos de comunicación de datos

Tabla 3.1: Especificaciones de Bluetooth 5.0, Wi-Fi y LoRa

3.1.2. Capas del protocolo LoRaWAN

El protocolo LoRaWAN está formado por 3 capas: la capa física, la capa *Medium Access Control* (MAC) y la capa de aplicación, las cuales se van a tratar en este apartado.[30]

Cada capa añade una funcionalidad, cuando una trama de datos se envía, esta se va encapsulando en cada capa del nivel inferior. La capa de aplicación normalmente está formada por los datos que se quieren enviar, estos suelen estar encriptados. Los servicios de la capa de aplicación son: enviar tramas

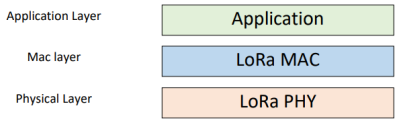


Figura 3.2: Capas del protocolo LoRaWAN [30]

a un grupo de puntos finales, enviar un bloque de datos fragmentados a puntos finales, mandar tramas para la sincronización de relojes y para la determinación del firmware del dispositivo final.[30]

La capa MAC está formada por una cabecera de 1 byte que define el tipo de mensaje (solicitud, aceptación, carga o descarga de datos...), la dirección del dispositivo (que viene dada por 4 bytes), un byte de control que contiene información sobre ADR, el contador de trama que está definido por 2 bytes, un campo de opción de trama que está definido de 0 a 15 bytes, el puerto de la aplicación, los datos enviados por el usuario y por último por el MIC, formado por 4 bytes.[30]

La capa física está formada por un preámbulo, una cabecera opcional que está activa en el modo de transmisión predeterminado e indica el tamaño de los datos y la velocidad de codificación, la información de la capa MAC y 2 bytes de *Cyclic Redundancy Check* (CRC) para detectar errores. En la figura 3.3 se puede observar como está formada una trama LoRaWAN con la información que añaden sus distintas capas[30]:

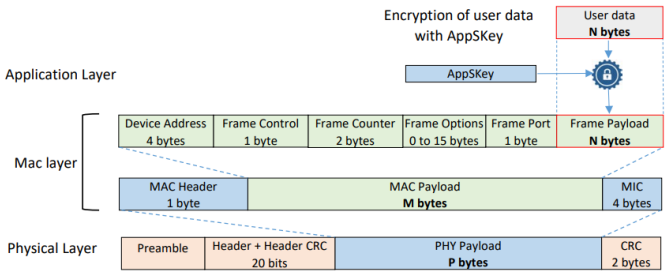


Figura 3.3: Trama LoRaWAN [30]

3.1.3. Arquitectura y seguridad de la red LoRaWAN

La arquitectura de una red LoRaWAN es en estrella, haciendo así que aumente la duración de la batería de los dispositivos, la capacidad y que sea

más simple, al contrario que sucede en otras redes que utilizan una arquitecura en forma de malla. Los nodos están conectados a múltiples pasarelas simultáneamente. Las pasarelas reenvían los datos que le llegan de los nodos a los servidores a través de un *backhaul* (Internet, Wi-Fi, satélite...). En los nodos están los sensores o las aplicaciones de control y motorización, como por ejemplo un sensor de humo o un medidor de agua. Los nodos móviles no se traspasan de un *gateway* a otro. El servidor de red gestiona la red, filtra los mensajes duplicados, realiza los controles de seguridad, optimiza la red y envía los datos al servidor de apicación [32]. En la figura 3.4 se muestra la arquitectura de una red LoRaWAN:

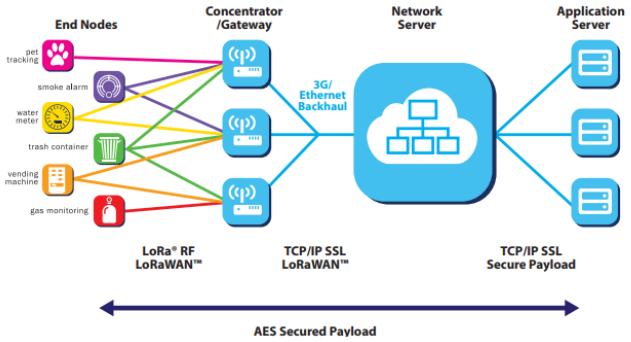


Figura 3.4: Arquitectura de una red LoRaWAN [29]

Entre el nodo y el servidor de la red, los mensajes se autentificán mediante una NwkSKey; y entre el nodo y el servidor de aplicación los mensajes están encriptados mediante una AppSKey. Cuando se envía una trama, esta primero se encripta mediante una encriptación simétrica con la AppSKey y después, a esta trama se le añaden los parámetros para la autenficación. En la autentificación, se añade el campo *Message Integrity Control* (MIC) a las tramas que se envían, este campo se forma dependiendo de la trama encriptada y de la NwkSKey. Cuando la trama llega al servidor de la red, si la NwkSKey es la misma que en el nodo, el MIC debe ser el mismo. Cuando la trama llega al servidor de aplicación, como se usa una encriptación simétrica, la AppSKey en este debe ser la misma que con la que se cifró en el mensaje [30]. A continuación, se muestran dos imágenes para que se entienda mejor:

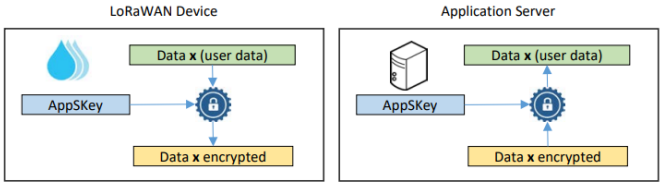


Figura 3.5: Encriptado de una trama [30]

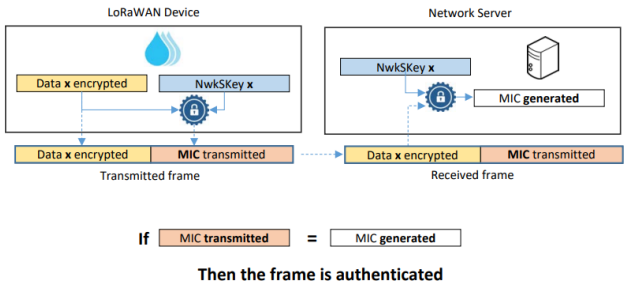


Figura 3.6: Autentificación de una trama [30]

Además de la NwkSKey y la AppSKey, también es esencial en la comunicación el DevAddr que identifica el nodo. Hay dos formas de proporcionar estos tres elementos al dispositivo final y al servidor: *Activation By Personalization* (ABP) y *Over the Air Activation* (OTAA). En ABP, el DevAddr, la NwkSKey y la AppSKey se guardan en el dispositivo final y en el servidor de forma manual por parte del administrador de la red[32]. En OTAA, la activación se realiza mediante el intercambio de mensajes *Join-Request* y *Join-Accept* entre el nodo y el servidor de red. Para ello el nodo tiene que tener configurado el DevEUI para identificar el dispositivo, el AppSKey para autentificar el mensaje *Join-Request*, encriptar el mensaje *Join-Accept* y generar las llaves de la sesión de comunicación; y el *AppEUI/JoinEUI* para identificar la aplicación. Cuando llega el mensaje *Join-Request* al servidor este genera las llaves NwkSKey y AppSKey y le manda el mensaje *Join-Accept* al dispositivo final y este genera las mismas llaves. El servidor de red a su vez envía la llave de aplicación al servidor de aplicación [30].

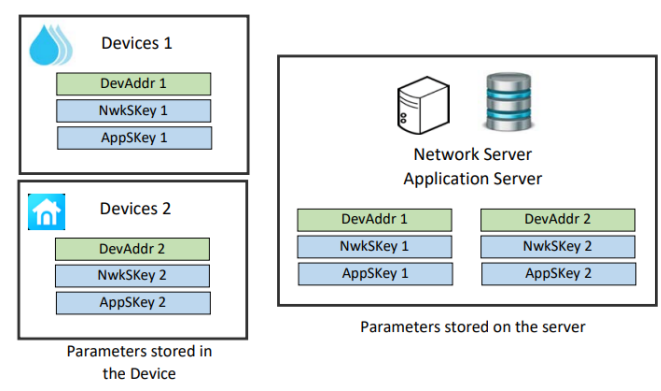


Figura 3.7: Activación mediante ABP [30]

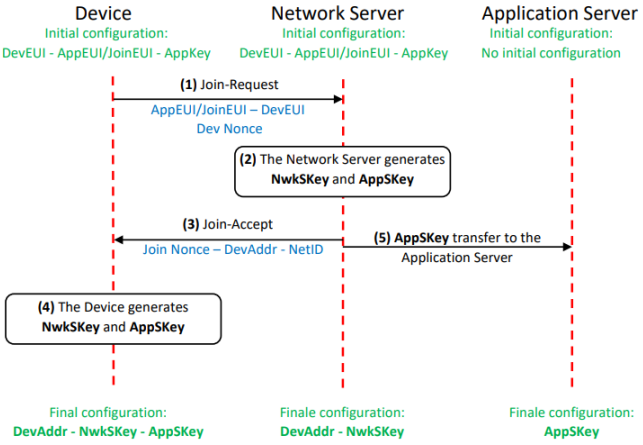


Figura 3.8: Activación mediante OTAA [30]

3.1.4. Clases de dispositivos

Los dispositivos finales están clasificados en 3 clases, según la duración de su batería y su capacidad de recibir mensajes de downlink [29]:

- **Clase A**: los dispositivos que pertenecen a esta clase permiten una comunicación bidireccional, en que la transmisión uplink viene seguida por dos ventanas de recepción downlink. Las ranuras de transmisión se basan en su propia necesidad de comunicación y una pequeña variación de tiempo aleatoria basada en el protocolo ALOHA. Esta clase de dispositivo es la que menos potencia consume y está diseñada para

comunicaciones en las que se necesitan mensajes de bajada después de un mensaje de subida ya que para que el servidor pueda enviar un mensaje tiene que esperar a que el dispositivo haya mandado antes uno.

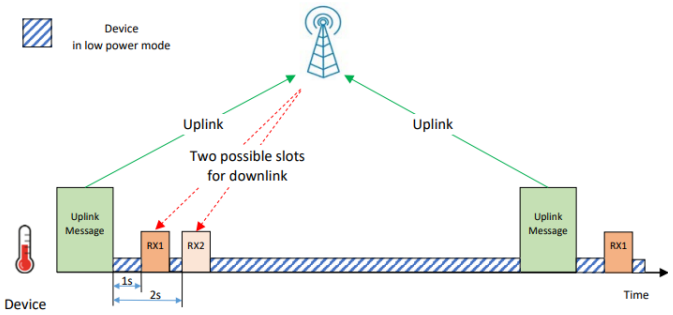


Figura 3.9: Comunicación de un dispositivo de clase A [30]

- **Clase B**: los dispositivos de esta clase además de recibir los mensajes en las ventanas aleatorias en las que reciben los dispositivos de clase A, también reciben en ventanas programadas. Para que el dispositivo final abra su ventana de recepción en el tiempo progamado, el *gateway* le manda un mensaje *beacon* de sincronización de tiempo, lo que hace que el servidor sepa cuando está esperando un mensaje *downlink*.

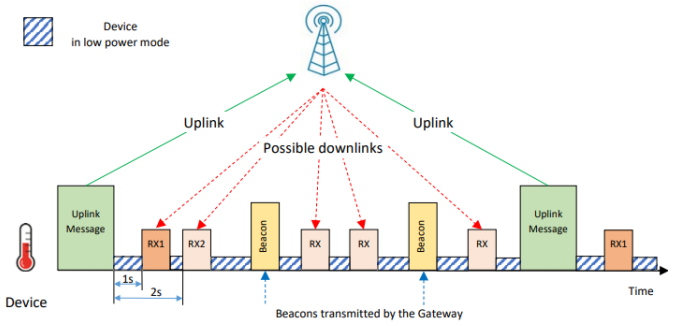


Figura 3.10: Comunicación de un dispositivo de clase B [30]

- **Clase C**: los dispositivos de clase C, tienen las ventanas de recepción abiertas siempre, lo que hace que sean los que menos latencia de *downlink* tengan pero también los que tienen menor duración de batería.

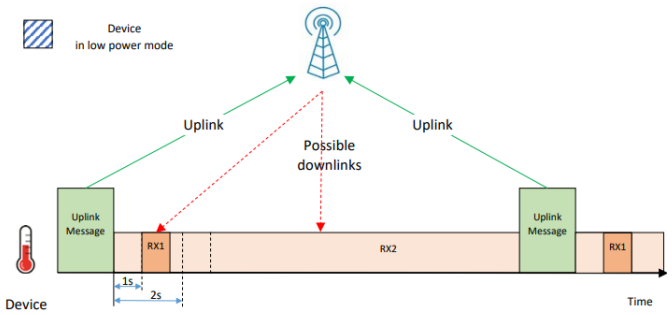


Figura 3.11: Comunicación de un dispositivo de clase C [30]

3.1.5. Message Queing Telemetry Transport

MQTT es un protocolo basado en publicación/suscripción muy usado en la tecnología IoT, en el que los publicadores envían mensajes a los distintos temas y los suscriptores reciben mensajes de los temas a los que están suscritos. Este protocolo se usa frecuentemente en LoRaWAN para obtener datos enviados por los sensores, ya que este protocolo está diseñado para dispositivos usados en aplicaciones de telemetría basado en TCP/IP. Está compuesto por dos entidades, los clientes y un *broker* que hace de intermediario entre los clientes que se suscriben o publican a distintos temas [34]. En la comunicación se intercambian mensajes de control, hay 14 tipos y están formados por 3 partes: una cabecera fija que está presente en todos, una cabecera variable que está fija en algunos tipos y el payload presentes en algunos tipos también [35]. Los tipos de mensajes son los siguientes [36]:

1. ***CONNECT***: el cliente envía este mensaje al *broker* para conectarse a él.
2. ***CONNACK***: el *broker* se lo envía al cliente para confirmar la solicitud de conexión.
3. ***DISCONNECT***: el cliente envía este mensaje al *broker* para desconectarse.
4. ***PINGREQ***: este mensaje es recibido o enviado por un cliente para comprobar que está conectado, es un mensaje *Keep Alive*.
5. ***PINGRESP***: es la respuesta al mensaje *PINGREQ* para confirmar que sigue conectado.
6. ***PUBLISH***: es un mensaje que un cliente envía al *broker* cuando hace una publicación sobre un tema e indica el tema. Este tipo de mensaje también es usado por el *broker* para enviar la publicación a los suscriptores de ese tema.

7. ***PUBACK***: es un mensaje en respuesta al mensaje *PUBLISH*. Puede ser enviado tanto por el *broker* como por un cliente suscrito a un tema.

8. ***PUBREC***: es un mensaje en respuesta al mensaje *PUBLISH* para indicar que este ha sido recibido correctamente. Puede ser enviado tanto por el *broker* como por un cliente suscrito a un tema.

9. ***PUBREL***: es un mensaje en respuesta al mensaje *PUBREC* para indicar que la publicación ha sido liberada. Puede ser enviado tanto por el *broker* como por un cliente publicador.

10. ***PUBCOMP***: es un mensaje en respuesta al mensaje *PUBREL* para indicar que la publicación ha sido completada. Puede ser enviado tanto por el *broker* como por un suscriptor.

11. ***SUBSCRIBE***: es un mensaje que envía un cliente al *broker* para indicar que quiere suscribirse a un tema, en el también indica el tipo de *Quality of Service* (QoS) que quiere utilizar.

12. ***SUBACK***: es una respuesta al mensaje *SUBSCRIBE* que envía el *broker* al cliente para confirmar su suscripción.

13. ***UNSUBSCRIBE***: es un mensaje que envía un cliente al *broker* para indicar que quiere desuscribirse de un tema.

14. ***UNSUBACK***: es una respuesta al mensaje *UNSUBSCRIBE* que envía el *broker* a los publicadores y suscriptores para confirmar su desuscripción.

El protocolo soporta tres niveles de QoS en función de la seguridad de la distribución de los mensajes:

- **QoS0**: el mensaje se envía como máximo una vez y no se garantiza que llegue a su destinatario.

- **QoS1**: el mensaje se envía al menos una vez y es posible que se envíe más veces marcando el valor de la *flag* que lleva el mensaje a 1.

- **QoS2**: el mensaje se envía una sola vez utilizando el método de comunicación *4-way handshaking*. En la figura 3.12 se muestra un ejemplo de este método de comunicación:

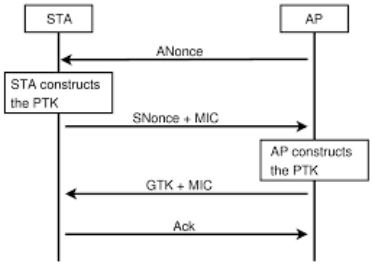


Figura 3.12: Ejemplo de *4-way handshaking* [37]

Los clientes pueden publicar mensajes, suscribirse a temas sobre los que quiere recibir mensajes o desuscribirse y desconectarse del *broker*. Por otro lado, el *broker* es el encargado de recibir y filtrar los mensajes y distribuirlos a sus correspondientes destinatarios; además reciben y gestionan las solicitudes de suscripción y desuscripción de los clientes. El *broker*, también retiene el último mensaje enviado por un publicador después de ser trasmitido a todos los clientes que están suscritos a los mensajes sobre ese tema, de forma que cuando llega un nuevo suscriptor de ese tema éste le envía los mensajes [35]. A continuación, se muestra un ejemplo de la comunicación entre clientes y el *broker* en la figura 3.13:

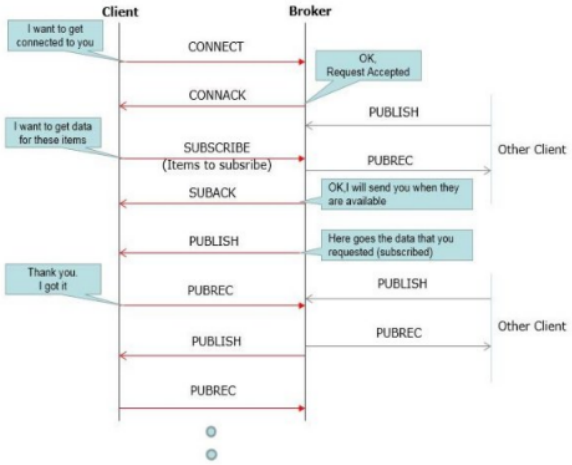


Figura 3.13: Ejemplo de comunicación del protocolo MQTT [35]

3.2. Red 5G

3.2.1. Descripción y escenarios en 5G

La red 5G es la quinta generación de comunicación a distancia móvil, tiene el propósito de aumentar la velocidad a la que se envía datos en la

red, la escalabilidad, la conectividad y la eficiencia de los dispositivos. Esto beneficia de forma importante el desarrollo de IoT haciendo que nos podamos comunicar con las máquinas y que las máquinas se comuniquen entre sí a mayor velocidad y también mejora el envío de datos entre personas [38].

Los principales servicios que se ofrecen son [39]:

- ***enhance Mobile Broadband*** **(eMBB)**: aborda casos de uso en el que el ser humano accede a contenido multimedia, servicio, datos, etcétera. Pudiendo conectarse a una velocidad de hasta 10 Gbps. Tiene una función importante por ejemplo en videollamadas o uso de realidad virtual.
- ***Ultra-Reliable and Low Latency Communications*** **(URLLC)**: aborda los escenarios en los que hay altos requisitos de rendiemiento, latencia, capacidad y disponibilidad. Las comunicaciones pueden llegar a tener una latencia de 1 ms. Esto es fundamental por ejemplo para contactar con servicios de emergencia, ad-hoc o manejo de maquinaria a distancia.
- ***massive Machine-Type Communications*** **(mMTC)**: para escenarios en los que se comunican entre sí un gran número de máquinas con poco retardo y paquetes de datos pequeños. Puede llegar a soportar un millon de conexiones por kilómetro cuadrado.

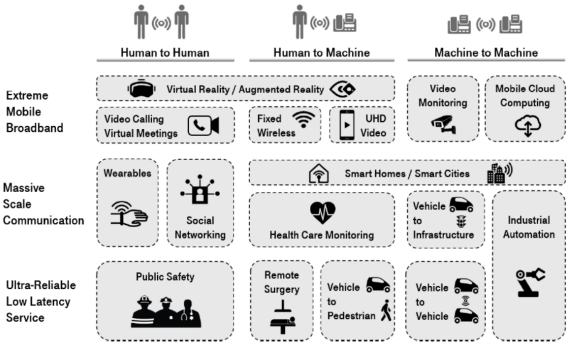


Figura 3.14: Escenarios de uso de 5G [40]

3.2.2. 3GPP

Third Generation Partnership Project (3GPP) [41] es una colaboración entre asociaciones cuyo principal objetivo, cuando se fundó en 1998, era fijar unas especificaciones técnicas e informes técnicos para el sistema móvil *Third*

Generation Mobile Communication (3G) basado en un core *Global System for Mobile Communication* (GSM) y las redes de acceso que soporta.

Más tarde, se puso como objetivo también mantener y desarrollar las tecnologías más allá del 3G. Actualmente, tienen como misión desarrollar la tecnología de las redes, especialmente para que soporten IoT de forma que las comunicaciones tengan una baja latencia, rapidez y un bajo coste energético. Quieren maximizar la compatibilad de los equipos que cumplen sus especificaciones con el core 5G, garantizando un sistema que puede soportar un gran número de casos de uso de los dispositivos y que cumplan con las expectativas de mercado y necesidades actuales.

3.2.3. Arquitectura de la red 5G

La arquitectura de la red 5G tiene como objetivo dar soporte a la conectividad para el envío de datos y distintos tipos de servicios como la virtualización. Sus características principales son la separación del plano de control y de usuario, permitir la escalabilidad de la red, permitir la creación de túneles en la red, garantizar la autentificación de los datos enviados y proporcionar una gran capacidad y baja latencia. Se divide principalmente en dos partes [42]:

- Red de acceso: cumple con la función de conectar los dispositivos con el core de la red 5G, 5GC. Esta red se puede conectar con el core a través de dos formas:

  - ***Radio Access Network* (RAN)**: se encarga de conectar los dispositivos que cumplen con las especificaciones establecidas por 3GPP. Los conecta a través de un gNB que puede tener conexiones con el CP, conectándose con el *Access and Mobility Management Function* (AMF), con el UP, conectándose con el *User Plane Function* (UPF).
  - ***Non-3GPP Interworking Function* (N3IWF)**: se encarga de conectar los dispositivos que se conectan a través de redes Non-3GPP, es decir, que no cumplen con las especificaciones establecidas por 3GPP como por ejemplo las redes Wi-Fi, BlueTooth o LoRaWAN.

- ***5G Core Network* (5GC)**: su función es conectar una red de acceso con otra red de acceso. Utiliza una SBA para garantizar la seguridad y la autenticación, gestionar las sesiones y el tráfico. Sus componentes son los siguientes:

  - ***Access and Mobility Management Function* (AMF)**: se conecta a la RAN o el N3IWF y con el equipo de usuario. Se

encarga de gestionar el registro, la conexión, el alcance y la movilidad de los dispositivos de usuario o *User Equipment* (UE), de su autenticación y de autorizar el acceso.

- ***Session Management Function* (SMF)**: se encarga principalmente de gestionar la sesión; por ejemplo, su establecimiento, las modificaciones que se hacen en esta o su finalización. Asigna y gestiona las direcciones IP de los UE, las cuales pueden llegar desde el UPF o desde una red externa, y configura el tráfico en el UPF para hacer un correcto enrutamiento de los paquetes que se transmiten. Se comunica con el UPF usando el protocolo *Packet Forwarding Control Protocol* (PFCP
- ***User Plane Function* (UPF)**: asigna las direcciones IP a los UE después de que el SMF se lo solicite. Es el punto de interconexión con la *Data Network* (DN). Se encarga del enrutamiento y reenvío de paquetes, de la inspección de paquetes, de hacer informes sobre el tráfico y del QoS en el plano de usuario.
- ***Policy Control Function* (PCF)**: contiene las políticas que regulan el comportamiento de la red, controla que se apliquen las políticas correctamente en el plano de control y accede a la información relevante sobre las decisiones de las políticas de la red en un *Unified Data Repository* (UDR).
- ***Network Exposure Function* (NEF)**: su función es exponer de forma segura las capacidades y eventos de la red, transduce la información entre el interior y el exterior de la red.
- ***Network Repository Function* (NRF)**: se encarga de gestionar los servicios NF, de aceptar su registro, mantenerlos, cancelar los registros y descubrirlos.
- ***Unified Data Management* (UDM)**: se encarga de crear las credenciales de autenticación, es la base de datos de los suscriptores y se encarga de controlarla. También gestiona los *Short Message Service* (SMS).
- ***Authentication Server Function* (AUSF)**: se encarga de la autenticación de los dispositivos de la red.
- ***Application Function* (AF)**: su función es proporcionar servicios de aplicación comunicándose con el NEF y cumpliendo las políticas de control.
- ***Network Slice Selection Function* (NSSF)**: su función es seleccionar el conjunto de instancias de *Network Slices* para servir a los dispositivos de usuario.

En la figura 3.15 se representa la arquitectura de la red 5G. En ella, se puede observar las distintas formas de acceso al core en función de si se usan

dispositivos que cumplen con las especificaciones establecidas por 3GPP o no, y que componentes de la red pertenecen al CP y al UP y como se comunican entre sí.

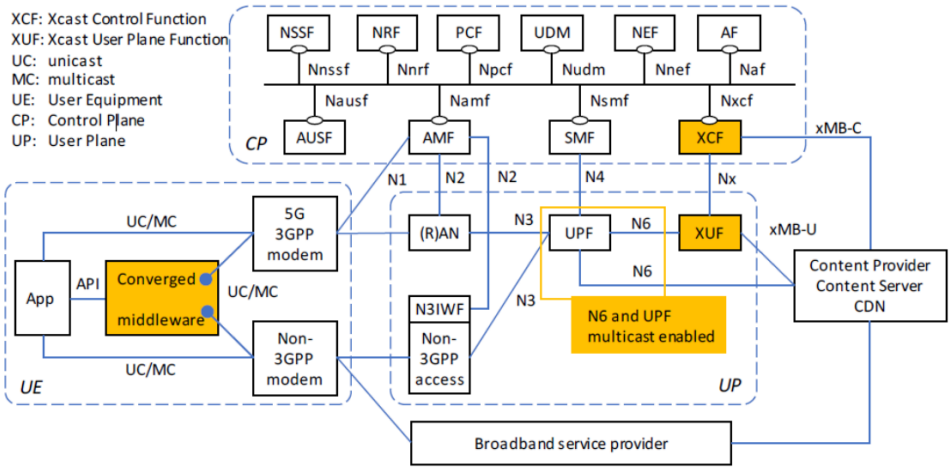


Figura 3.15: Arquitectura de la red 5G [40]

El despliegue de la arquitectura se puede hacer de dos formas [43]:

- ***Non-Stand Alone* (NSA)**: en este tipo de despliegue, la red de acceso de 5G y su interfaz NR se utilizan en conjunto con la red 4G, es decir, no se sustituye la red completamente. En este despliegue, no se pueden disfrutar de todos los servicios de 5G, pero si de las capacidades que ofrece su interfaz NR

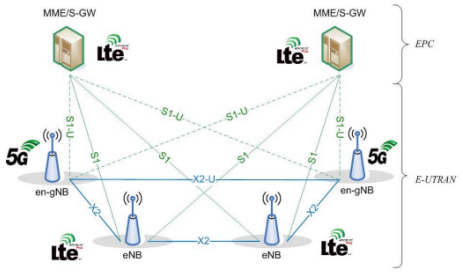


Figura 3.16: Arquitectura de la red NSA [43]

- ***Stand Alone* (SA)**: en este otro tipo de despliegue, la interfaz NR está conectada al 5GC. De esta forma, se pueden soportar todos los servicios que tienen como objetivo ofrecerse en la tecnología 5G, es decir, eMBB, URLLC y mMTC.

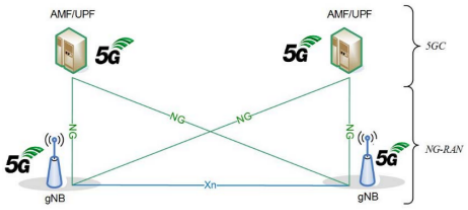


Figura 3.17: Arquitectura de la red SA [43]

Capítulo 4

Planificación y coste

En este capítulo se mostrará la planificación temporal del proyecto; los recursos software, hardware y humanos empleados en él; y finalmente una estimación del coste.

4.1. Planificación temporal

La duración total del proyecto ha sido de 182 días. La planificación inicial para su realización es la siguiente:

1. **Planteamiento y Presentación del Proyecto**:
    - **Planteamiento del Proyecto (11 días)**: periodo en el que se reunen tutor y alumno para plantear el TFG.
    - **Propuesta y Aceptación del Proyecto (9 días)**: periodo desde que se propone el TFG hasta que es aceptado oficialmente.
2. **Estudio y Análisis**
    - **Estudio del Estado del Arte (15 días)**: estudio de las distintas herramientas para la implementación y la integración de las redes LoRaWAN y 5G, y selección de cuál es la más adecuada. Además, recopilación y análisis de información sobre artículos y contribuiciones científicas relacionadas con el proyecto.
    - **Estudio de la red LoRaWAN (15 días)**: recopilación y estudio sobre la red LoRaWAN, la modulación, LoRa; los protocolos y su arquitectura.
    - **Estudio de la red 5G (15 días)**: estudio y análisis de los casos de uso de la red 5G, sus innovaciones y mejoras respecto de otras tecnologías anteriores, y el estudio de su arquitectura y la función de los elementos que la componen.

3. **Diseño e Implementación**

   - **Diseño de la Red (20 días)**: análisis de las diferentes opciones para la implementación de las redes LoRaWAN y 5G. Así como el diseño del modelo de arquitectura para la integración de ambas redes.
   - **Diseño e Implementación de las Motas (13 días)**: proceso de configuración y programación de las motas TTGO LORA32-OLED.
   - **Diseño e Implementación del *Gateway* para la red LoRaWAN (13 días)**: proceso de configuración y programación del *gateway Pygate*.
   - **Diseño e Implementación de los Servidores LoRaWAN (14 días)**: instalación del software para implementar los servidores LoRaWAN y su configuración para la red.
   - **Diseño e Implementación del Core 5G (18 días)**: instalación del software para implementar el core 5G y su configuración para la red.
   - **Diseño e Implementación del cliente NWu (7 días)**: instalación del software para implementar el cliente NWu y su configuración para la red.
   - **Integración de las Redes LoRaWAN y 5G y el cliente NWu (7 días)**: diseño y configuración de tablas de enrutamiento para integración de las redes LoRaWAN y 5G y el cliente NWu.

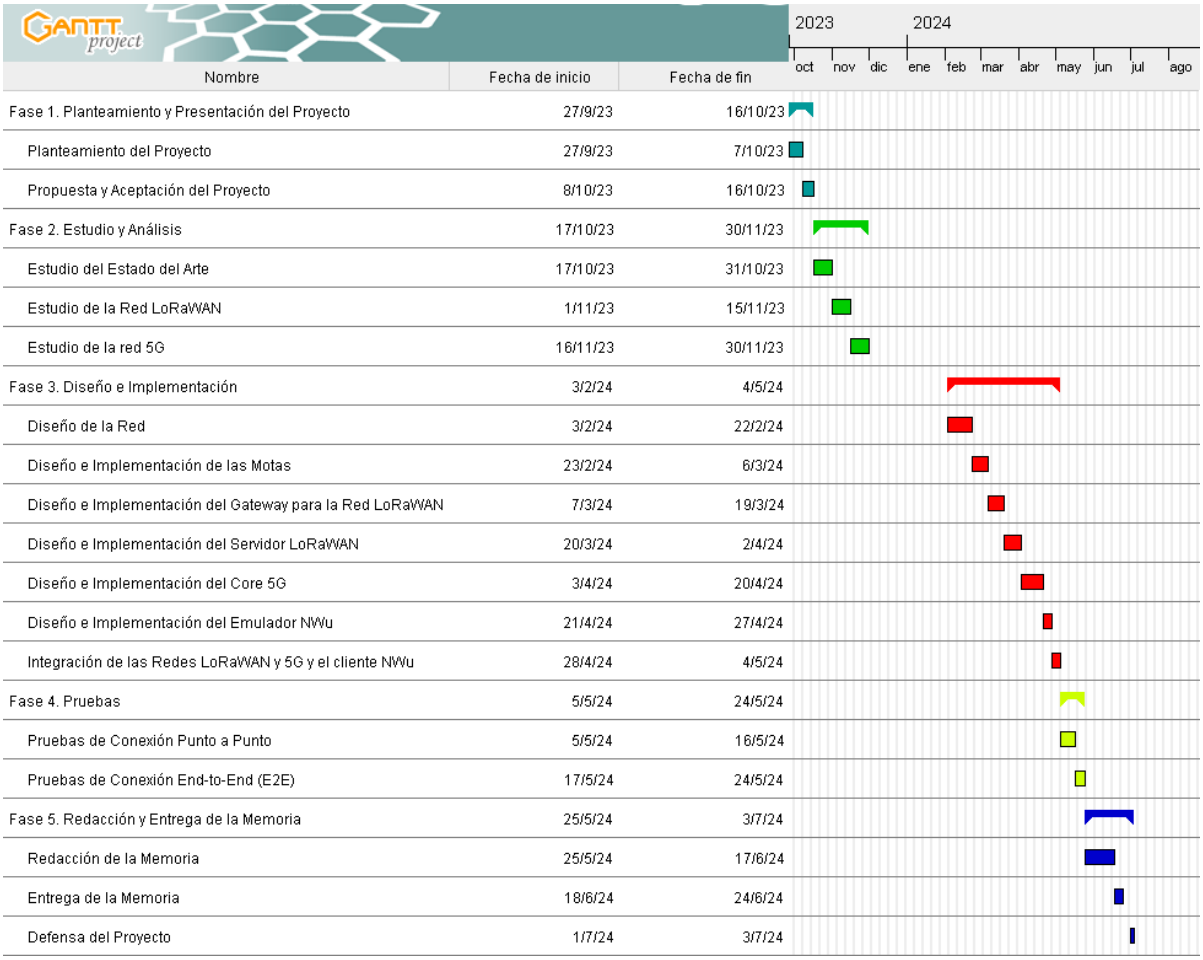
4. **Pruebas (20 días)**:

   - **Pruebas de conexión punto a punto (12 días)**: periodo en el que se realizan pruebas de conexión entre los distintos saltos dentro de la red diseñada.
   - **Pruebas de conexión *End-to-End* (E2E) (8 días)**: periodo en el que se realizan pruebas de conexión entre la mota y el servidor LoRaWAN, ambos son los puntos finales de la red.

5. **Redacción y Entrega de la Memoria**:

   - **Redacción de la Memoria (24 días)**: redacción completa del proyecto, de forma lógica y estructurada.
   - **Entrega de la Memoria (7 días)**: envío de la solicitud de la defensa del proyecto y entrega de su memoria.
   - **Defensa del Proyecto (3 días)**: defensa del proyecto ante el Tribunal asignado.

En la figura 4.1 se muestra un diagrama de Gantt y una tabla. En ellos, se ve esta planificación a lo largo del tiempo y se distinguen las distintas fases.



Nombre	Fecha de inicio	Fecha de fin
Fase 1. Planteamiento y Presentación del Proyecto	27/9/23	16/10/23
Planteamiento del Proyecto	27/9/23	7/10/23
Propuesta y Aceptación del Proyecto	8/10/23	16/10/23
Fase 2. Estudio y Análisis	17/10/23	30/11/23
Estudio del Estado del Arte	17/10/23	31/10/23
Estudio de la Red LoRaWAN	1/11/23	15/11/23
Estudio de la red 5G	16/11/23	30/11/23
Fase 3. Diseño e Implementación	3/2/24	4/5/24
Diseño de la Red	3/2/24	22/2/24
Diseño e Implementación de las Motas	23/2/24	6/3/24
Diseño e Implementación del Gateway para la Red LoRaWAN	7/3/24	19/3/24
Diseño e Implementación del Servidor LoRaWAN	20/3/24	2/4/24
Diseño e Implementación del Core 5G	3/4/24	20/4/24
Diseño e Implementación del Emulador NWu	21/4/24	27/4/24
Integración de las Redes LoRaWAN y 5G y el cliente NWu	28/4/24	4/5/24
Fase 4. Pruebas	5/5/24	24/5/24
Pruebas de Conexión Punto a Punto	5/5/24	16/5/24
Pruebas de Conexión End-to-End (E2E)	17/5/24	24/5/24
Fase 5. Redacción y Entrega de la Memoria	25/5/24	3/7/24
Redacción de la Memoria	25/5/24	17/6/24
Entrega de la Memoria	18/6/24	24/6/24
Defensa del Proyecto	1/7/24	3/7/24

Figura 4.1: Diagrama de Gantt

4.2. Recursos y costes

En esta sección, se exponen los recursos utilizados para realizar el proyecto y sus respectivos costes. Está dividida en recursos hardware, software y humanos; y el coste total.

4.2.1. Recursos software

Los recursos software empleados para la realización del proyecto son los siguientes. Considerar que se ha pretendido, en la medida de lo posible, emplear recursos *open source*:

- **Sistema Operativo Windows 10 Pro** [44]: es el sistema operativo instalado en el ordenador portatil en el que se encuentran las máquinas virtuales para la realización del proyecto y otras herramientas software.

- **Arduino IDE** [45]: es el entorno de desarrollo que se utilizará para programar las placas de las motas TTGO LORA32-OLED, las cuales utilizan una board ESP32. En el se usarán los paquetes: "*Adafruit GFX*", "*Adafruit BusIO*", "*ESP8266 and ESP32 OLED driver*", "*IBM LMIC frameworkz* "*LoRa*". La versión utilizada es la 2.3.2.

- **Visual Studio Code** [46]: es un editor de código fuente ligero desarrollado por Microsoft. Será la herramienta utilizada para programar el gateway. En él, se usará el paquete *Pymkr* que tiene el lenguaje MicroPython; y la versión 1.89.

- **Oracle VM VirtualBox** [47]: es un software de virtualización x86 y AMD64/Intel64. En él, se montarán las máquinas virtuales que se utilizarán a lo largo del proyecto. Se usará la versión 7.0.18.

- **Ubuntu** [48]: es un software libre y de código abierto distribuido por GNU/Linux. Las versiones utilizadas son la 22.04 para implementar la red LoRaWAN y la 20.04 para implementar la red 5G.

- **Docker Compose** [49]: es una aplicación para definir y ejecutar aplicaciones de servidores múltiples. Simplifica el control de la aplicación haciendo sencilla su configuración y permitiendo ejecutarla introduciendo un único comando. Permite iniciar, reiniciar, parar servicios y ver su estado.

- **Chirpstack** [7]: es un servidor de red y aplicación LoRaWAN de código abierto que se puede utilizar para configurar redes LoRaWAN públicas o privadas. Proporciona una interfaz web para gestionar la red. Este servidor va a ser instalado y ejecutado utilizando Docker Compose.

- **Emulador NWu** [22]: es un emulador NWu realizado en python3, que hace que se pueda conectar el UE con el 5GC a través de él estableciendo túneles IKEv2/IPSec con el N3IWF o con el *evolved Packet Data Gateway* (ePDG).

- **Free5gc** [17]: es un proyecto de código abierto que implemnta el core de la red 5G definida en 3GPP Release 15 (R15). Los mayores contribuidores de este proyecto son de la National Yang Ming Chiao Tung University.

- **Wireshark** [50]: es un software gratuito que sirve para analizar y solucionar problemas en una red de comunicación y sus protocolos y los datos enviados.

- **Overleaf** [51]: es un editor LaTex colaborativo basado en la nube. Permite escribir, editar, visualizar y publicar documentos de forma online en tiempo real. Es el editor utilizado para la redacción de la memoria del proyecto.

- **GanttProject** [52]: es un programa de código abierto con licencia GPL escrito en Java, que permite crear diagramas de Gantt. La versión utilizada en el proyecto es la 3.3.

A continuación, en la tabla 4.1 se representan los costes de los recursos software, para el sistema operativo instalado en el ordenador se realiza una amortización a tres años que se estima que tenga de vida, teniendo en cuenta que su coste fué de 168,99 €:

Software	**Coste (€)**
Sistema Operativo Windows 10 Pro	28,17
Arduino IDE	0
Visual Studio Code	0
Oracle VM VirtualBox	0
Ubuntu	0
Docker Compose	0
Emulador NWu	0
Free5gc	0
Chirpstack	0
Wireshark	0
Overleaf	0
GanttProject	0
Coste Total	**28,17**

Tabla 4.1: Costes de Recursos Software

4.2.2. Recursos hardware

Los recursos hardware empleados en el proyecto son los siguientes:

- **Ordenador portátil** [53]: herramienta principal del proyecto en la que se alojan los recursos software. En la tabla 4.2 se muestran sus principales características:

- **Mota TTGO LORA32-OLED**: dispositivo empleado para el envío de mensajes cada 20 segundos (aproximádamente) de forma automática.

Modelo	Lenovo ThinkPad E15 GEN2 20TDCTO1WW
Sistema Operativo	Windows 10 Pro
Procesador	Intel® Core™ i7-1165G7 de cuádruple núcleo a 2,8GHz y Turbo Boost de hasta 4,7GHz
Memoria RAM	32 GB DDR4 a 3.200MHz SoDIMM
Almacenamiento	512 GB M.2 2242 SSD
Tarjeta Gráfica	Intel® Iris® Xe

Tabla 4.2: Especificaciones Técnicas del Portátil



Figura 4.2: Mota TTGO LORA32-OLED

- **Pygate** [54]: dispositivo que cumple la función de *gateway* entre la mota y el servidor. Es un *gateway* LoRaWAN de 8 canales desarrollo por Pycom.



Figura 4.3: Pygate

En la siguiente tabla se presentan los costes de los recursos hardware, estimando que el tiempo de vida del ordenador portátil es de 3 años y que su coste de adquisición fue de 785,57 €.

Recurso Hardware	**Unidades**	**Coste/Unidad (€)**	**Coste Total (€)**
Ordenador portátil	1	130,93	130,93
Mota TTGO LORA32-OLED	1	14,20	14,20
Pygate	1	98,99	98,99
Coste Total			**244,12**

Tabla 4.3: Costes de Recursos Hardware

4.2.3. Recursos humanos

Los recursos humanos involucrados en el proyecto son las personas que han trabajado en él. Estas personas son:

- **Jorge Navarro Ortiz**: Profesor Titular de Universidad del Área de Ingeniería Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada. Tutor del proyecto.
- **Félix Delgado Ferro**: Investigador Predoctoral en el Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada. Cotutor del proyecto.
- **Juan Guillermo García-Valdecasas Mendoza**: Estudiante del Grado de Ingeniería de Tecnologías de Telecomunicación de la Universidad de Granada. Autor del proyecto.

En la siguiente tabla se muestra una estimación de los costes de recursos humanos:

Persona	**Precio/hora (€)**	**Horas Totales**	**Precio Total (€)**
Jorge Navarro Ortiz	50	30	1500
Félix Delgado Ferro	35	20	700
Juan Guillermo García-Valdecasas Mendoza	20	350	7000
Precio Total			**9200**

Tabla 4.4: Coste de Recursos Humanos

4.2.4. Coste total

El coste total del proyecto se muestra resumido en la siguiente tabla:

Recurso	**Coste (€)**
Recursos Software	28,17
Recursos Hardware	244,12
Recursos Humanos	9200
Coste Total	9472,29

Tabla 4.5: Costes Totales

El coste total del proyecto es de 9472,29 €, NUEVE MIL CUATROCIENTOS SETENTA Y DOS EUROS Y VEINTINUEVE CÉNTIMOS.

Capítulo 5

Diseño e implementación

En este apartado, se va a explicar cómo se ha realizado el diseño y la implementación de la integración de las red LoRaWAN y 5G paso a paso.

5.1. Diseño del modelo de arquitectura para la Integración de LoRaWAN y 5G

El propósito principal del proyecto es crear una red en la que dispositivos LoRaWAN puedan enviar mensajes a sus servidores de red y de aplicación a través de un *core* 5G. En el diseño, se van a analizar las funciones de cada elemento que compone la red. En la figura 5.1 se presenta la arquitectura de la red diseñada.

- **Mota**: es un dispositivo móvil que se encarga de enviar mensajes LoRa cada 20 segundos aproximádamente simulando un sistema que envía información cada cierto tiempo al servidor, como puede ser un dispositivo en un invernadero inteligente.
- ***Gateway***: se encarga de recibir los mensajes transmitidos por la mota, vía LoRa, y enviarlos al cliente NWu a través de Wi-Fi.
- **Cliente NWu**: se encarga de iniciar una sesión segura con el N3IWF del core 5G creando un túnel entre ellos mediante IKEv2/IPSec y el proceso de autenticación con el core 5G mediante 5G-AKA. Una vez creado este túnel, envía los mensajes que le llegan del *gateway* a través del túnel al N3IWF.
- ***Core* 5G**: su función es reenviar los mensajes que le llegan al N3IWF a los servidores de red y de aplicación de LoRaWAN. Esto lo hace autentificando previamente al cliente NWu, lo realiza el AMF, y creando una sesión con él, esto lo hace el SMF. Después envía el tráfico del N3IWF al UPF, a través de un túnel GTP que crea entre los dos. Y, finalmente, el UPF lo envía al servidor LoRaWAN.

- **Plataforma LoRaWAN**: recibe los mensajes en el puerto 1700, correspondiente al servidor de red, y los reenvía a su servidor de aplicación correspondiente. En este caso, se crean los servidores de red y aplicación con ChirpSatck. Esta permite visualizar los mensajes que llegan y su contenido.

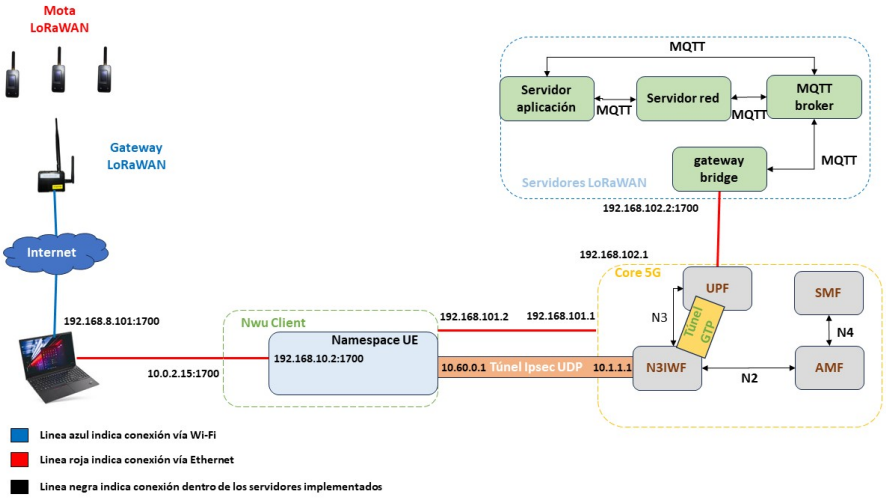


Figura 5.1: Diseño de la Arquitectura para la Integración de las redes LoRaWAN y 5G

5.2. Implementación y Configuración de las Motas

Las motas tienen como objetivo el envío de mensajes LoRa cada 20 segundos aproximádamente de forma que lleguen a su correspondiente aplicación sin tener un alto consumo. Para ello, se han utilizado motas TTGO LORA32-OLED.

Para su configuración de software, se ha utilizado la aplicación Arduino IDE, con los paquetes que se comentan en la sección 4.2.1; y, el lenguaje de programación C++. El código utilizado para la configuración puede encontrarse en [55]. Este código, permite definir *Spreading Factor* (SF), el espectro de frecuencia, los intervalos de transmisión, los datos que se envían; y, las claves NwkSKey y AppSKey para la sesión LoRaWAN. Además, cuenta con una lista de dispositivos para identificar cuál es el que está enviando el mensaje y distintas funciones de inicialización en función de la red a la que esté conectada. De igual modo, configura una pantalla incorporada a la mota, en la que se puede visualizar: el número de mensajes enviados, el instante

de envío del siguiente mensaje, el número de identifiación del dispositivo, la versión del dispositivo y si le sesión se configura mediante ABP o OTAA.

En el caso de este proyecto, el SF configurado en la mota es el SF7. Se envía los mensajes en la frecuencia 868 MHz, y mantiene un ciclo de trabajo donde se transmite cada 20 segundos, como se ha mencionado anteriormente. Los datos que se envían varían en función del número del mensaje y del canal. En la figura 5.2 se muestran las NwkSKey y AppSKey y los identificadores de las distintas motas. En este caso, la mota es la número 12.

```
static deviceDataStruct devicesData[noDevices] = {
  { 1, 0xBD3C8E2DE6B4, 0x00000065, "0000000000000065", "TTGO LORA32"},
  { 2, 0x79458E2DE6B4, 0x00000066, "0000000000000066", "TTGO LORA32"},
  { 3, 0x09408E2DE6B4, 0x00000067, "0000000000000067", "TTGO LORA32"},
  { 4, 0x51458E2DE6B4, 0x00000068, "0000000000000068", "TTGO LORA32"},
  { 5, 0xB93F8E2DE6B4, 0x00000069, "0000000000000069", "TTGO LORA32"},
  { 6, 0x993C8E2DE6B4, 0x0000006A, "000000000000006A", "TTGO LORA32"},
  { 7, 0x5CFA73A4AE30, 0x0000006B, "000000000000006B", "TTGO LORA32"},
  { 8, 0x9D3C8E2DE6B4, 0x0000006C, "000000000000006C", "TTGO LORA32"},
  { 9, 0x09458E2DE6B4, 0x0000006D, "000000000000006D", "TTGO LORA32"},
  {10, 0xA93C8E2DE6B4, 0x0000006E, "000000000000006E", "TTGO LORA32"},
  {11, 0x71438E2DE6B4, 0x0000006F, "000000000000006F", "TTGO LORA32"},
  {12, 0xE93F8E2DE6B4, 0x00000070, "0000000000000070", "TTGO LORA32"},
  {13, 0x0D458E2DE6B4, 0x00000071, "0000000000000071", "TTGO LORA32"},
  {14, 0xC1DD8E2DE6B4, 0x00000072, "0000000000000072", "TTGO LORA32"},
  {15, 0xADDD8E2DE6B4, 0x00000073, "0000000000000073", "TTGO LORA32"},
  {16, 0x45438E2DE6B4, 0x00000074, "0000000000000074", "TTGO LORA32"},
  {17, 0xA4D143A4AE30, 0x00000075, "0000000000000075", "TTGO LORA32"},
  {18, 0xED3F8E2DE6B4, 0x00000076, "0000000000000076", "TTGO LORA32"},
  {19, 0xF53D8E2DE6B4, 0x00000077, "0000000000000077", "TTGO LORA32"},
  {20, 0xE13F8E2DE6B4, 0x00000078, "0000000000000078", "HELTEC WIRELESS STICK"},
  {21, 0xD096FE7EB994, 0x000000C9, "00000000000000C9", "HELTEC WIRELESS STICK"},
  {22, 0x0896FE7EB994, 0x000000CA, "00000000000000CA", "PYCOM FIPY"},
  {23, 0x9D41C33A7D80, 0x0000012D, "000000000000012D", "PYCOM FIPY"}
};
deviceDataStruct deviceData;

static const PROGMEM u1_t NWKSKEY[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01 };
static const u1_t PROGMEM APPSKEY[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01 };
```

Figura 5.2: Configuración motas [55]

En las figuras 5.3 y 5.4, se muestra el envío de mensajes de la mota visto desde Arduino IDE (los mensajes *here* es para la identificación de posibles errores en el código) y desde la pantalla de la propia mota, respectivamente. Estos mensajes los envía cifrados. En la imagen 5.4, también se pueden ver los datos citados antes (identifiador de dispositivo, configuración ABP o OTAA y la versión del dispositivo).

```
[INFO] Data length: 10
Packet queued
here(6)here(5)here(6)here(5)here(6)here(5)here(6)here
Time between last two packets:      19.11 sec
Time between enqueuing and sending: 2.10 sec
Time to wait for next frame:        17.90 sec
```

Figura 5.3: Envío del mensaje de una mota visto desde Arduino [55]



Figura 5.4: Información de la pantalla de la mota [55]

5.3. Implementación y Configuración del *Gateway*

El *gateway*, tiene que recibir los mensajes de la mota y enviarlos al cliente NWu vía Wi-Fi. El *gateway* empleado es un Pygate, que puede utilizar la tecnología LoRaWAN y 8 canales.

Para configurarlo, se emplea un código del repositorio [55], la aplicación Visual Studio Code y el lenguaje MicroPython. Este código, permite configurar una IP para el servidor LoRaWAN de destino. En este caso, será la dirección IP del ordenador personal en la red Wi-Fi. Y, configura el *gateway* para que detecte la red Wi-Fi, se conecte a ella y envíe los mensajes a la IP de destino. Para ello, hay que indicar el nombre de la red y su contraseña. Además, al igual que el código de la mota, cuenta con una lista de *gateways* para identificar cuál es cada uno, en este caso el 8. Cabe mencionar, que se configura el router Wi-Fi para que asigne una IP fija al ordenador personal cuando detecte su dirección MAC. De este modo, la configuración de la IP del servidor LoRaWAN que se configura en el *gateway* es siempre la misma.

En la figura 5.5, se puede observar como detecta la red Wi-Fi y se conecta a ella, también se observa el identificador del dispositivo y como se inician los SF.

```
Starting LoRaWAN concentrator
Gateway EUI: 807d3afffe948778
Device tag:  PYGATE08

Found MOVISTAR_FE1E network!
Connecting to WiFi..... OK
Wi-Fi network configuration: ('192.168.8.105', '255.255.255.0', '192.168.8.1', '192.168.8.1')

Syncing RTC via ntp..... OK

buf: {
    "SX1301_conf": {
      "lorawan_public": true,
      "clksrc": 1,
      "antenna_gain": 0,
      "radio_0": {
        "enable": true,
        "type": "SX1257",
        "freq": 867500000,
        "rssi_offset": -166,
        "tx_enable": true,
        "tx_freq_min": 863000000,
        "tx_freq_max": 870000000
      },
      "radio_1": {
        "enable": true,
        "type": "SX1257",
        "freq": 868500000,
        "rssi_offset": -166,
        "tx_enable": false
      },
      "chan_multiSF_0": {
        "enable": true,
        "radio": 1,
        "if": -400000
```

Figura 5.5: Inicialización del *gateway*, detección de la red Wi-Fi [55]

Y, en la figura 5.6, se pueden observar como se han iniciado los distintos canales del gateway (recuadro rojo), y como está configurado para enviar los mensajes que recibe al puerto 1700 de la dirección IP del ordenador personal, 192.168.8.101 (recuadro amarillo). Después de esta inicialización, queda a la espera de recibir mensajes.

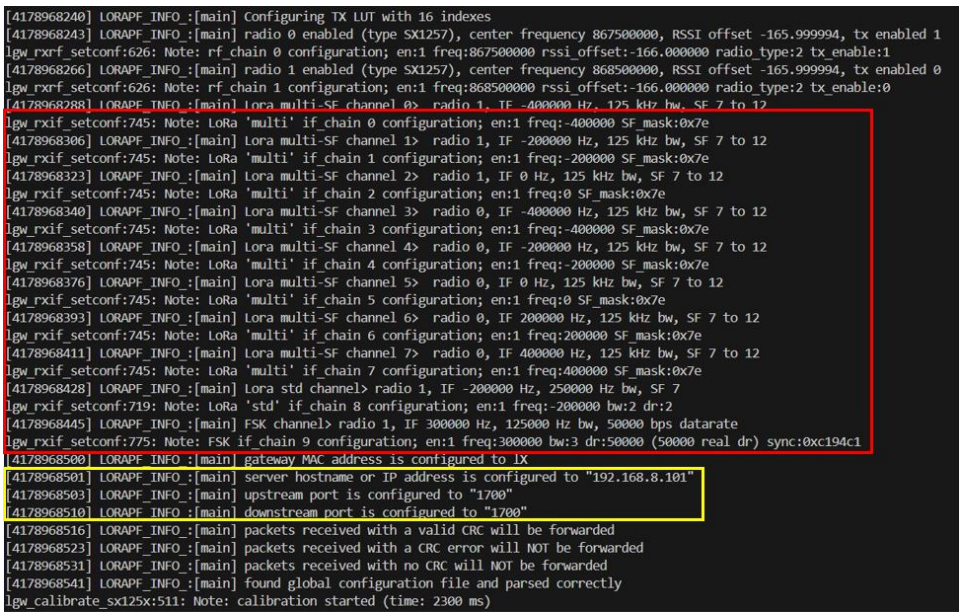


Figura 5.6: Inicialización del *gateway*, canales y IP destino [55]

5.4. Implementación y Configuración de la Plataforma LoRaWAN

El servidor de LoRaWAN, se implementa haciendo uso de ChirpStack, como se explica en 2.4. Esta implementación, se hace en una máquina virtual usando Oracle VM VirtualBox con el software Ubuntu 22.04. Esta tiene asignada una interfaz en modo red interna con la IP 192.168.102.2, para luego conectarse con el *core* 5G. El objetivo de ChirpStack, es recibir los mensajes cifrados que envían las motas y que han pasado por el *gateway*, el cliente NWu y por el *core* 5G; descrifrar estos mensajes y mostrarlos en la aplicación correspondiente. A lo largo de toda la red, estos mensajes mantienen su integridad.

A continuación, se van a explicar los pasos para instalación y la configuración del servidor LoRaWAN:

1. **Intalación y configuración de ChirpStack**: en primer lugar, hay que instalar y configurar ChirpStack y su *broker* MQTT como se explica en el apéndice B. Una vez hecho esto, se accede a la interfaz web en la dirección `http://localhost:8080` con el usuario *admin* y la clave *admin*.

2. **Creación de una organización**: en el apartado *tenant* de la interfaz web se crea una organización. En ella, se incluirán los distintos

gateways y aplicaciones.

3. **Creación de *gateways***: dentro de la organización creada, en el apartado *Gateways*, se añaden las pasarelas que componen la red. Se añade el nombre con el que se quiere identificar a cada uno, y su identificador de dispositivo en *Gateway ID*.

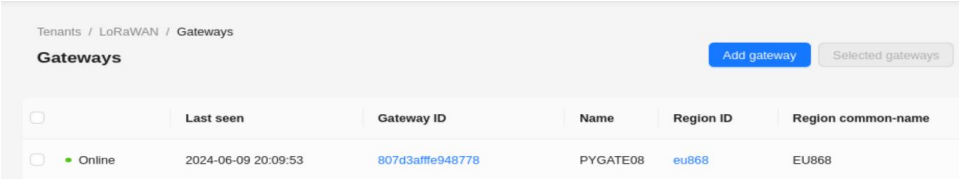


Figura 5.7: Creación de *gateways* en ChirpStack

4. **Creación de perfil de dispositivo**: en la sección *Device Profiles*, dentro de la organización creada, se crean los perfiles de los dispositivos. Cuando se crea un perfil de dispositivo, hay que añadir el nombre con el que se va a identificar, la región y frecuencia en la que está, en este caso EU868; la versión MAC del dispositivo, el algoritmo ADR, en este caso el que viene por defecto (solo de LoRa) y, añadiendo el intervalo de tiempo máximo esperado de *uplink* en segundos. En este caso, la activación es mediante ABP, por lo que no se configura la activación mediante OTAA en el dispositivo.

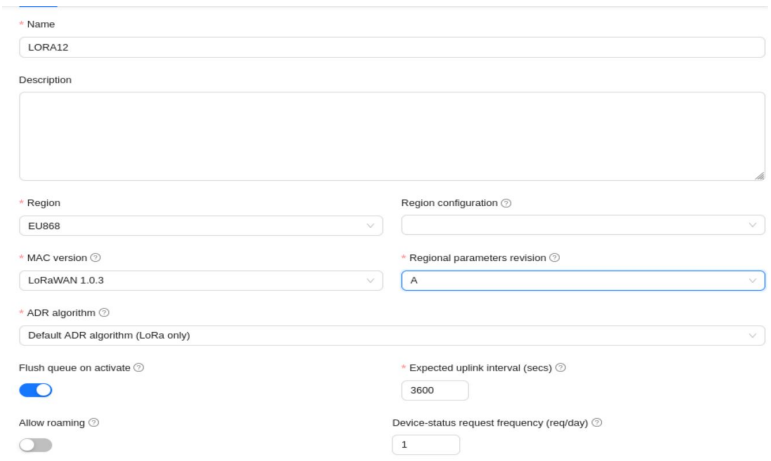


Figura 5.8: Creación de perfil de dispositivo en ChirpStack

5. **Creación de aplicación**: también dentro de la organización creada, se crea la aplicación, en el apartado *Applications*, con el nombre con el que se quiera identificar. Dentro de la aplicación creada, se añaden

los equipos, indicando su nombre, el *device Extended Unique Identifier 64 bits* (EUI64) y el *join* EUI64, se elige el perfil del usuario de ese dispositivo y, se deshabilita la validación por contador de trama. Mientras, en el apartado de *Activation* del equipo añadido, se indica su dirección y su NwkSKey y AppSKey. En las figuras 5.9 y 5.10 se muestra la configuración del dispositivo dentro de la aplicación.

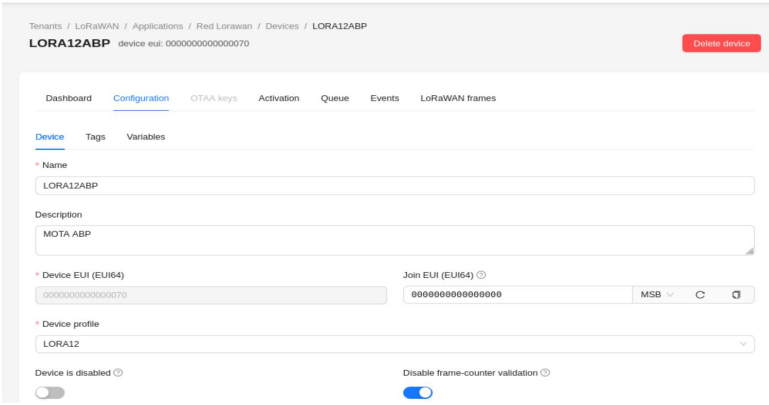


Figura 5.9: Configuración de dispositivo en aplicación de ChirpStack

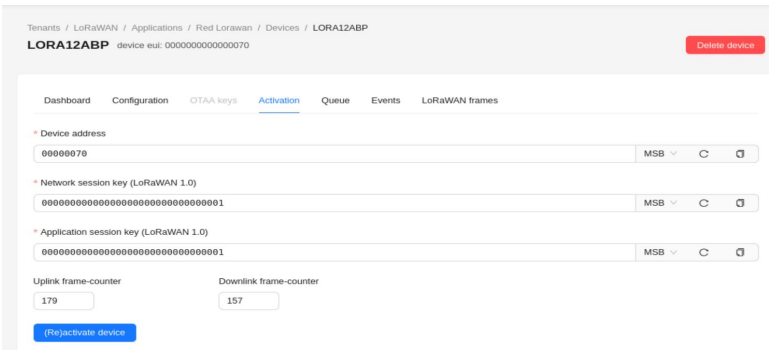


Figura 5.10: Configuración de dispositivo en aplicación de ChirpStack (activación)

Una vez configurado todo correctamente, se pueden comprobar los dispositivos y *gateways* activos en la red y el tráfico que les llega:

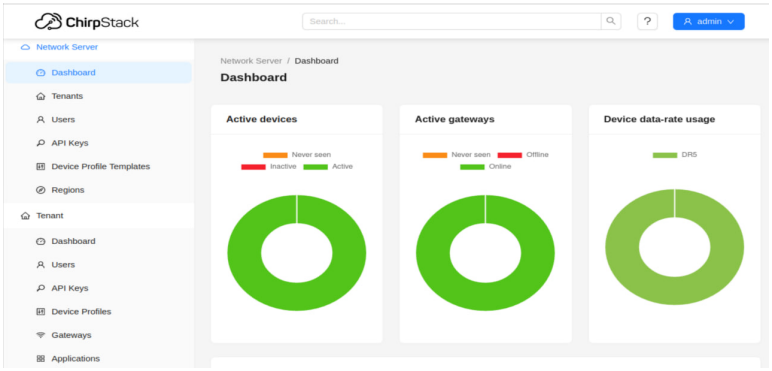


Figura 5.11: Comprobación de la Plataforma LoRaWAN en ChirpStack

5.5. Implementación y Configuración del *Core* 5G

El *core* 5G se implementa usando la herramienta free5gc. Este servidor también se implementa en una máquina virtual en Oracle VM VirtualBox, pero en este caso, con el software Ubuntu 20.04. La máquina virtual tiene dos interfaces de red interna: la primera interfaz con la IP 192.168.102.1, para conectarse con la máquina en la que se implementa el servidor LoRaWAN y, la segunda interfaz con la IP 192.168.101.1, para conectarse con la máquina virtual en la que se va a implementar el cliente NWu. La finalidad de free5gc, en el proyecto, es recibir los paquetes de datos del cliente NWu y mandarlo al servidor LoRaWAN, implementando las funciones de los distintos elementos de un *core* 5G. Este mantiene en todo momento el cifrado y la integridad de estos mensajes.

Para la implementación y configuración del *core* 5G, en primer lugar, se realiza su instalación como se indica en apéndice C. Una vez hecha la instalación, se configuran los elementos de free5gc que realizan la función de los distintos elementos de 5GC. La mayoría de estos elementos ya están configurados correctamente, pero hay que comprobar que las direcciones IP que tiene configuradas son correctas. Para acceder a los archivos de configuración de cada uno de ellos, se usa el comando `sudo nano amfcfg.yaml` (ejemplo de configuración de AMF). Este y el resto de comandos se configuración se encuentran en la carpeta */free5gc/config_n3iwf*. A continuación, se explica la configuración de las direcciones IP de cada uno de los nodos que componen el *core* 5G, la explicación se centra solo en las IP ya que las funciones que debe realizar cada uno, las cuáles se explican en 3.2.3, vienen implementadas correctamente al instalar free5gc:

- **AMF**: en la sección *configuration*, en la lista de interfaces N2 (*ngapIpList*), es decir, de la interfaz de AMF que se conecta con N3IWF;

se introduce la IP que se le quiera dar a esta interfaz para conectarse con el N3IWF en la misma máquina virtual, en este caso 127.0.0.18; y, también se añade la IP de la interfaz de la red de la máquina virtual que se conecta con el cliente NWu, la 192.168.101.1. Además, se indica la IP de esta interfaz dentro de la misma máquina virtual, es decir, la 127.0.0.18; al NRF. Para ello, se escribe esta IP en *registerIPv4* y *bindinIPv4*. Del mismo modo, se indica el puerto en el que se quiere que funcione, en *port*; y la IP y el puerto del NRF en *nrfUri*. En la figura 5.12, se muestra la configuración.

```
configuration:
  amfName: AMF # the name of this AMF
  ngapIpList:  # the IP list of N2 interfaces on this AMF
    - 192.168.101.1 # For gNB (another machine)
    - 127.0.0.18    # For N3IWF (same machine)
  ngapPort: 38412 # the SCTP port listened by NGAP
  sbi: # Service-based interface information
    scheme: http # the protocol for sbi (http or https)
    registerIPv4: 127.0.0.18 # IP used to register to NRF
    bindingIPv4: 127.0.0.18  # IP used to bind the service
    port: 8000 # port used to bind the service
    tls: # the local path of TLS key
      pem: cert/amf.pem # AMF TLS Certificate
      key: cert/amf.key # AMF TLS Private key
  serviceNameList: # the SBI services provided by this AMF, refer to TS 29.518
    - namf-comm # Namf_Communication service
    - namf-evts # Namf_EventExposure service
    - namf-mt   # Namf_MT service
    - namf-loc  # Namf_Location service
    - namf-oam  # OAM service
  servedGuamiList: # Guami (Globally Unique AMF ID) list supported by this AMF
    # <GUAMI> = <MCC><MNC><AMF ID>
    - plmnId: # Public Land Mobile Network ID, <PLMN ID> = <MCC><MNC>
        mcc: 208 # Mobile Country Code (3 digits string, digit: 0~9)
        mnc: 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
      amfId: cafe00 # AMF identifier (3 bytes hex string, range: 000000~FFFFFF)
  supportTaiList:  # the TAI (Tracking Area Identifier) list supported by this AMF
    - plmnId: # Public Land Mobile Network ID, <PLMN ID> = <MCC><MNC>
        mcc: 208 # Mobile Country Code (3 digits string, digit: 0~9)
        mnc: 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
      tac: 000001 # Tracking Area Code (3 bytes hex string, range: 000000~FFFFFF)
  plmnSupportList: # the PLMNs (Public land mobile network) list supported by this AMF
    - plmnId: # Public Land Mobile Network ID, <PLMN ID> = <MCC><MNC>
        mcc: 208 # Mobile Country Code (3 digits string, digit: 0~9)
        mnc: 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
      snssaiList: # the S-NSSAI (Single Network Slice Selection Assistance Information) list supported by this AMF
        - sst: 1 # Slice/Service Type (uinteger, range: 0~255)
          sd: 010203 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
        - sst: 1 # Slice/Service Type (uinteger, range: 0~255)
          sd: 112233 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
  supportDnnList:  # the DNN (Data Network Name) list supported by this AMF
    - internet
  nrfUri: http://127.0.0.10:8000 # a valid URI of NRF
```

Figura 5.12: Configuración del AMF en free5gc

- **SMF**: en este elemento, se configura su dirección IP en *registerIPv4* y *bindinIPv4* en la sección *configuration*, con esta IP se va a registrar en el NRF, en este caso es la 127.0.0.2. Paralelamente, se comprueba la dirección de *Domain Name Server* (DNS), en la sección de *Single Network Slice Selection Assistance Information* (S-NSSAI), en este caso 8.8.8.8. Lo siguiente que tiene configurado es la dirección IP del UPF, 127.0.0.8, y la interfaz de conexión del SMF con el UPF, la 127.0.0.1; a través de esta interfaz tienen una comunicación utilizando el protocolo *Packet Forwarding Control Protocol* (PFCP) con la que gestionan las sesiones de datos. Por último, tiene configurada la IP del NRF. Esto se comprueba en en el apartado de cada elemento dentro de la sección *configuration*.

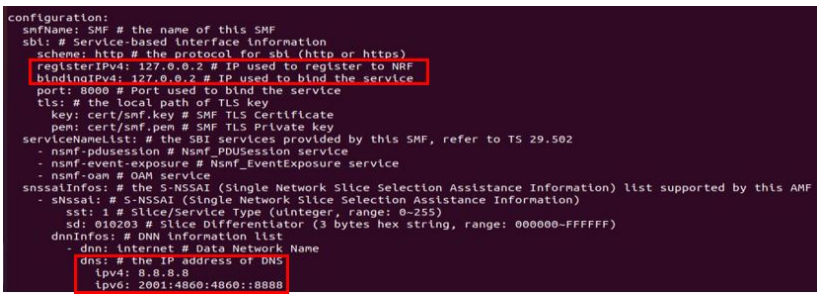


Figura 5.13: Configuración del SMF en free5gc

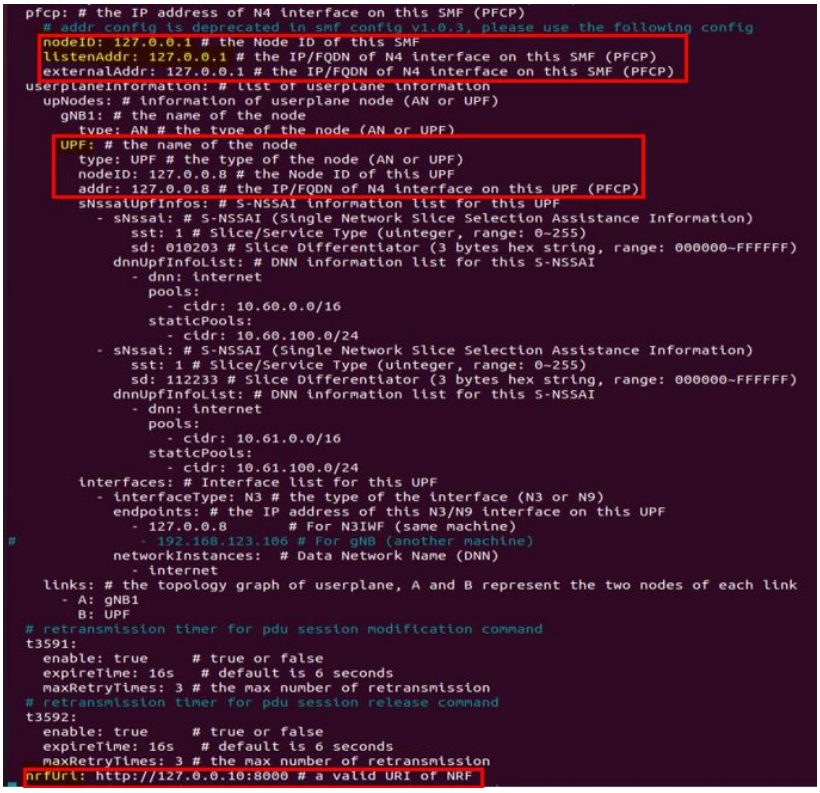


Figura 5.14: Configuración del SMF en free5gc

- **UPF**: en este elemento, se configura la dirección IP para comunicarse con el SMF y la dirección IP para el túnel GTP 5G. En ambas es la misma, la 127.0.0.8:

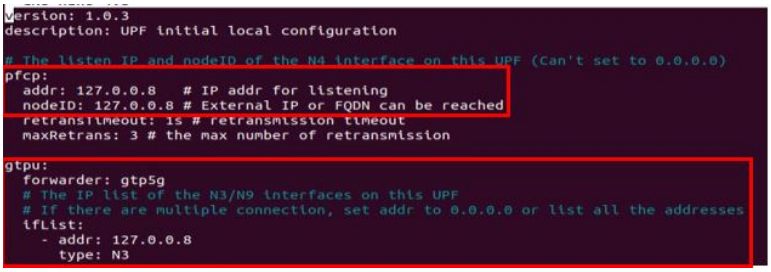


Figura 5.15: Configuración del UPF en free5gc

- **PCF**: en este elemento, se configura su dirección IP, la 127.0.0.7; y, la dirección IP del NRF para poder registrarse:

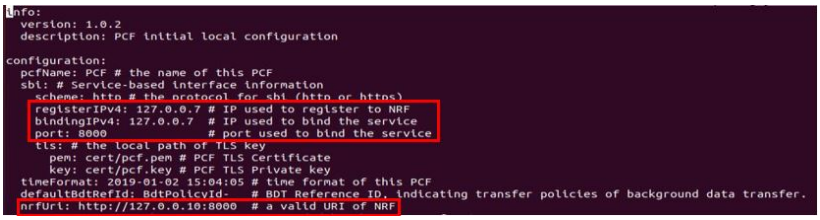


Figura 5.16: Configuración del PCF en free5gc

- **NRF**; en este elemento, se configura su dirección IP, y el puerto en el que escucha: 127.0.0.10:8000:

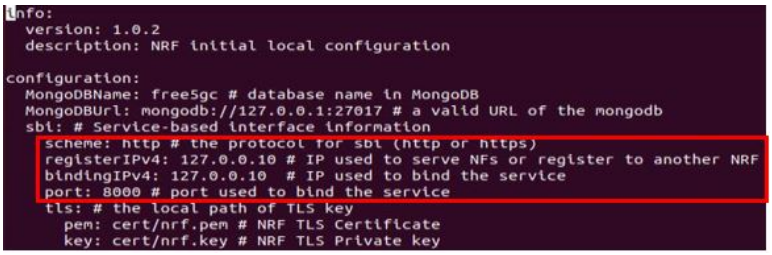


Figura 5.17: Configuración del NRF en free5gc

- **UDM**: en este elemento, se configura su dirección IP, en este caso la 127.0.0.3; y, la dirección IP del NRF:

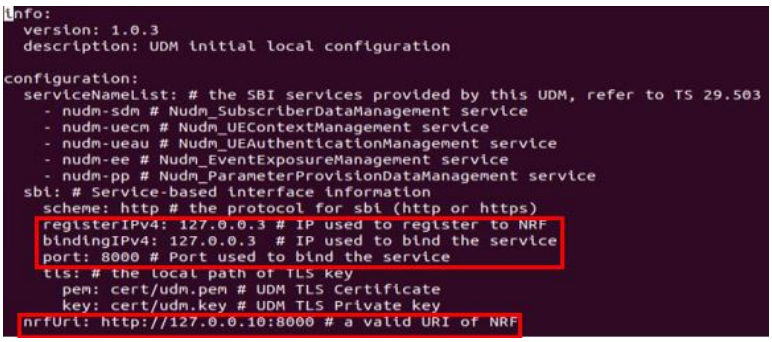


Figura 5.18: Configuración del UDM en free5gc

- **AUSF**: en este elemento, también se configura su dirección IP, en este caso la 127.0.0.9; y, la dirección IP del NRF:

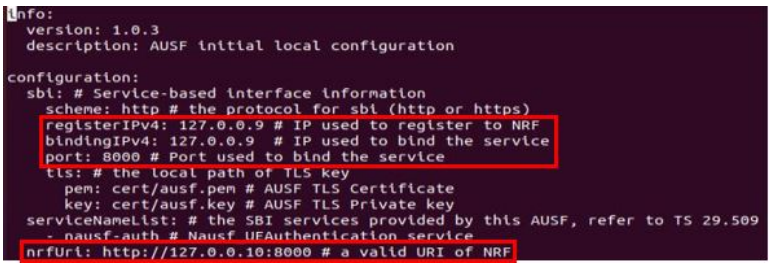


Figura 5.19: Configuración del AUSF en free5gc

- **NSSF**: en este elemento, al igual que en otros anteriores, también se configura su dirección IP, en este caso la 127.0.0.31; y, la dirección IP del NRF. Cabe mencionar, que en el proyecto no se ha llevaa cabo la modificación y configuración de las *Network Slices*:

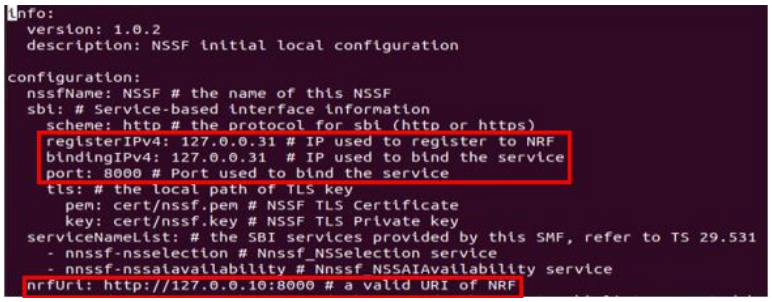


Figura 5.20: Configuración del NSSF en free5gc

- **UDR**: en este elemento, también, se configura su dirección IP, en este caso la 127.0.0.4; y, la dirección IP del NRF:

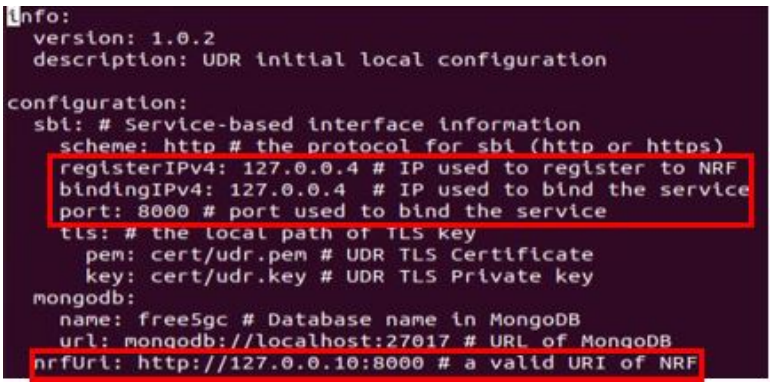


Figura 5.21: Configuración del UDR en free5gc

- **N3IWF**: este elemento es de suma importancia configurarlo correctamente, ya que es al que llegan los paquetes de datos desde el cliente N3IWF. En él, se configura su interfaz N2, al igual que en AMF. Es decir, la IP de la interfaz en la propia máquina virtual, 127.0.0.18; y, de la interfaz que se conecta con la máquina virtual donde está el cliente NWu, 192.168.101.1. En la sección de *Nwu Interfaces*, se configura la dirección IP para realizar la autentificación con el cliente NWu, en *IKEBindAddress*, 192.168.101.1. Además, se indica la dirección de la interfaz del N3IWF en el túnel que se crea cuando se conecta con el cliente NWu, a través del que se envían los paquetes de datos, 10.1.1.1. También, se indica, en la sección de *N3 Interfaces*, la direeción IP de la interfaz para el túnel GTP 5G, 127.0.0.33.

```
info:
  version: 1.0.5
  description: N3IWF initial local configuration

configuration:
  N3IWFInformation:
    GlobalN3IWFID: # ID used to globally identify an N3IWF
      PLMNID: # Public Land Mobile Network ID, <PLMN ID> = <MCC><MNC>
        MCC: 208 # Mobile Country Code (3 digits string, digit: 0~9)
        MNC: 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
      N3IWFID: 135 # ID used to identify an N3IWF in PLMN (uinteger, range: 0~65535)
    Name: free5GC_N3IWF # The name of this N3IWF
    SupportedTAList: # Tracking Area supported by this N3IWF
      - TAC: 000001 # Tracking Area Code (3 bytes hex string, range: 000000~FFFFFF)
        BroadcastPLMNList: # Refer to TS 38.413
          - PLMNID: # Public Land Mobile Network ID
              MCC: 208 # Mobile Country Code (3 digits string, digit: 0~9)
              MNC: 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
            TAISliceSupportList: # Network Slice supported in this TAI
              - SNSSAI: # Single Network Slice Selection Assistance Information
                  SST: 1 # Slice/Service Type (1 byte hex string, range: 0~F)
                  SD: 010203 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
              - SNSSAI: # Single Network Slice Selection Assistance Information
                  SST: 1 # Slice/Service Type (1 byte hex string, range: 0~F)
                  SD: 112233 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)

    # --- N2 Interfaces ---
    AMFSCTPAddresses: # the IP list of N2 interfaces (SCTP) on AMF when using NGAP
      - IP:
          - 127.0.0.18
          - 192.168.101.1
        Port: 38412
    NASTCPPort: 20000 # TCP port which the NAS listens on

    # --- Nwu Interfaces ---
    IKEBindAddress: 192.168.101.1   # Nwu interface  IP address (IKE) on this N3IWF
    IPSecTunnelAddress: 10.1.1.1    # Tunnel IP address of XFRM interface on this N3IWF
    UEIPAddressRange: 10.1.1.0/24   # IP address pool allocated to UE in IPSec tunnel
    XFRMInterfaceName: xfrmi        # Prefix of XFRM interface name created by N3IWF
    XFRMInterfaceID: 1              # XFRM interface if_id for IPSec routing (Any value except to 0, default value is 7 if not defined)

    # --- N3 Interfaces ---
    GTPBindAddress: 127.0.0.33 # IP address of N3 interface (GTP) on this N3IWF

    FQDN: n3iwf.free5gc.org # FQDN of this N3IWF
```

Figura 5.22: Configuración del N3IWF en free5gc

Cuando se ha realizado esta configuración, se arranca el *core* 5G ejecutando el archivo *jna_start_all_n3iwf* que se encuentra en la carpeta *free5gc*, con el comando `sudo ./jna_start_all_n3iwf`. Este archivo elimina procesos activos que pueden interferir en la iniciación del *core* 5G y, acto seguido inicia la puesta en marcha de los distintos elementos de free5gc. Una vez el servidor está arrancado, se lanza la interfaz web introduciendo el comando `go run server.go` en la carpeta */free5gc/webconsole*. A ella, se accede desde la dirección *http://localhost:5000*, y se introduce el usuario *admin* y la contraseña *free5gc*. Desde ella, se añaden los suscriptores a la red 5G introduciendo sus datos, en este caso, los del cliente NWu con el que se va a conectar:

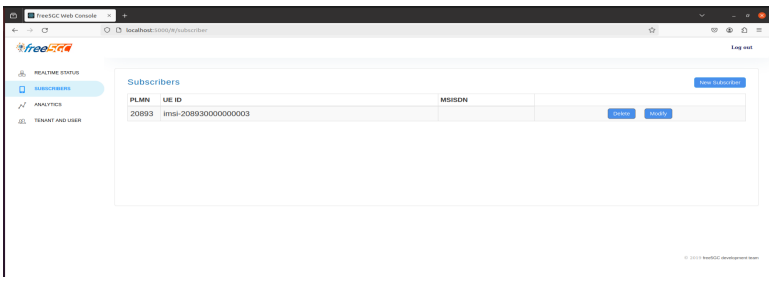


Figura 5.23: Interfaz web de free5gc

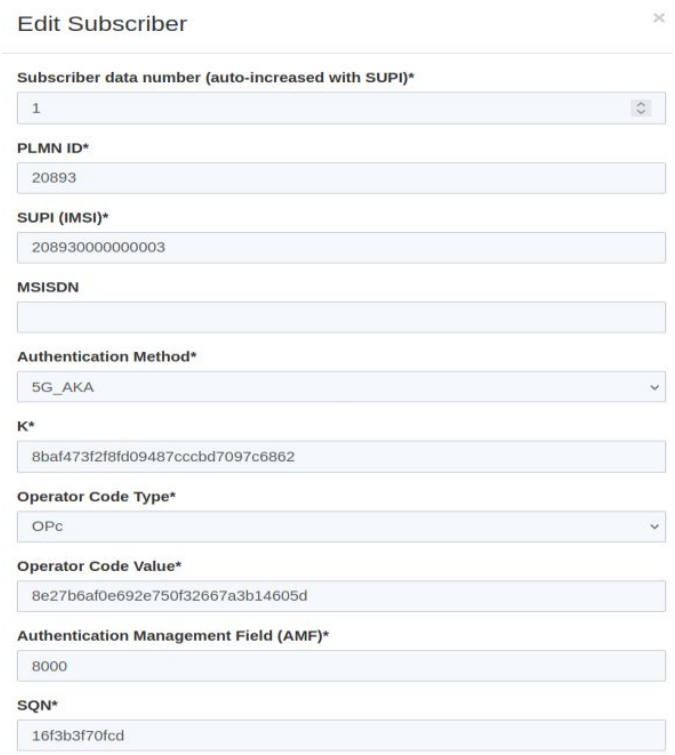


Figura 5.24: Configuración de suscriptor en free5gc

Finalmente, se hace una última configuración en la máquina virtual. Utilizando *iptables*, se configura el enrutamiento de los paquetes UDP que llegan al puerto 1700 de la interfaz del túnel GTP del N3IWF y los redirige al puerto 1700 de la máquina virtual en la que se implementa la plataforma LoRaWAN. Esto es necesario porque ChirpStack está configurado para recibir los paquetes de datos en este puerto. Para ello hay que configurar previamente el reenvío de paquetes en la máquina virtual con el comando `sudo sysctl -w net.ipv4.ip_forward=1`. Esta configuración de enrutamiento se realiza de la siguiente forma:

```
sudo iptables -t nat A PREROUTING -p udp --dport 1700 -j DNAT
    --to-destination 192.168.102.2:1700
```

Este comando redirige todo el tráfico UDP que llega al puerto 1700 hacia la IP de la máquina virtual donde se implementa la plataforma LoRaWAN, 192.168.102.2:1700. Introduciendo este comando, se le indica al UPF que tiene que redirigir el tráfico de datos de esta forma.

5.6. Implementación y Configuración del cliente *Networking Unit* (NWu)

El cliente NWu, tiene como objetivo iniciar una sesión segura con el N3IWF del 5GC, creando un túnel entre ellos y, recibir los mensajes del *gateway* y enviarlos al 5GC. Para implementarlo, se hace uso del repositorio [22]. Su funcionamineto se ha explicado en 2.3.1. La implementación se realiza, al igual que las anteriores, en una máquina virtual usando Oracle VM VirtualBox. En esta máquina, se utiliza el software Ubuntu 22.04.

Tiene configurada una interfaz interna de red para conectarse con la máquina en la que se encuentra el *core* 5G, con la IP 192.168.101.2. También tiene configurado el reenvío de puertos en el adaptador de red *Network Address Translation* (NAT), para que los paquetes UDP que lleguen al puerto 1700 del ordenador personal. Y así, se redirijan al puerto 1700 de la máquina virtual, se muestra en la figura 5.25:

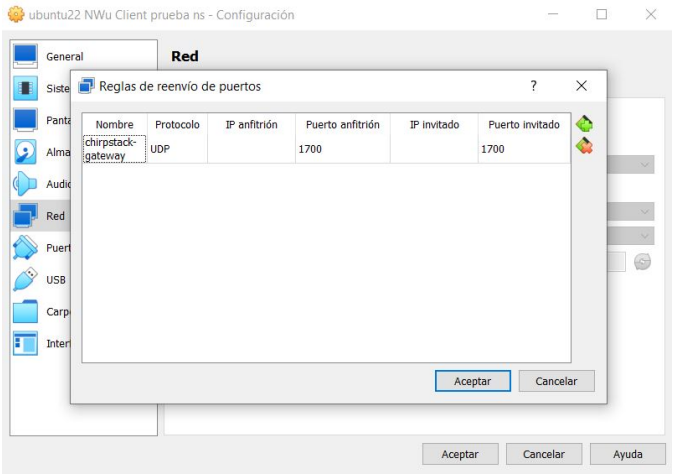


Figura 5.25: Configuración de reenvío de puertos en la máquina virtual del cliente NWu

La instalación del cliente NWu se explica en el apéndice D. Una vez se ha instalado, se configura el archivo *jna_connect_n3iwf_free5gc.sh*, dentro de la carpeta *NWu-Non3GPP-5GC client*. Este archivo arranca el cliente NWu con los parámetros que se configuran dentro de él. Dentro de este archivo, se especifica que si detecta la interfaz de red que se conecta con la máquina virtual del 5GC, en este caso *enp0s8*, que la IP donde está el N3IWF es 192.168.101.1. Además, se configuran los datos USIM que simulan un dispositivo móvil para la autenticación y conexión con free5gc. Los datos de USIM, son los mismos que los que se introducen en la interfaz web de free5gc cuando se agrega un suscriptor (véase en la sección 5.5). Esto se

muestra en la figura 5.26. Se han usado los archivos de configuración de NAS y de seguridad que figuran en el repositorio [22].

```
#!/bin/bash

# Main parameters
TYPE='NWU' # NWU (5G) or SWU (4G)
FIXFree5GCBUGS='Y'
USE_CA_CERTIFICATE_PUBKEY='N'
NAMESPACE='ue'

IF='enp0s8'
N3IWFADDR='192.168.101.1'
MTU=1300
TUNADDR='60.60.0.1'

# APN and slice parameters
APN='internet'
DEFAULT_SD_1='66051'   #'0x010203'
DEFAULT_SD_2='1122867' #'0x112233'
AMF_REGION_ID='202' # 0xCA
AMF_SET_ID='1016'   # 0x3F8
AMF_POINTER='0'

# USIM data
MCC='208'
MNC='93'
IMSI='208930000000003'  # Configured in core as EAP-AKA
#IMSI='208930000000004' # Configured in core as EAP-AKA'
KI='8baf473f2f8fd09487cccbd7097c6862'
OPC='8e27b6af0e692e750f32667a3b14605d'
```

Figura 5.26: Configuración de archivo para arrancar el cliente NWu

Al ejecutar este archivo, el cliente NWu se conecta con el N3IWF, haciendo primero un proceso de autenticación 5G-AKA y, luego, creando un túnel entre ellos. Se demostrará que se hace correctamente en el capítulo de pruebas 6.3. Cuando se crea el túnel, el cliente NWu lo hace desde un *namespace* llamado *ue*, que crea durante la conexión con el N3IWF. La interfaz del túnel del *namespace* tiene la IP 10.60.0.1/32.

Para configurar el enrutamiento de los paquetes de datos que llegan desde el *gateway* hacia free5gc, se ha creado un fichero ejecutable llamado *namespaceNWu.sh*. Este fichero, primero configura el reenvío de paquetes en la máquina virtual y crea una interfaz virtual para comunicarse con el *namespace*. La interfaz en el lado de la máquina virtual tiene la IP 192.168.10.1/24, *veth0*, y en el *namespace* la IP 192.168.10.2/24, *veth1*. Después de esto, se indican las rutas para llegar a la interfaz de la máquina virtual *namespace* y para llegar a la interfaz del túnel del *namespace* del cliente NWu con el N3IWF. Seguidamente, se hace el enrutamiento de los paquetes de datos UDP que llegan del *gateway* hacia el *namespace* y, del *namespace* hacia la interfaz del túnel de N3IWF, usando una máscara para que cuando se haga el seguimiento de los paquetes en la máquina del *core* 5G aparezca que la IP de origen es la del *namespace*. Este archivo, se muestra en la figura 5.27:

```
#!/bin/bash

#Configuracion reenvio de paquetes
sudo sysctl -w net.ipv4.ip_forward=1

#Crear interfaces virtuales
sudo ip link add veth0 type veth peer name veth1

#Configurar interfaces del namespace para conectar con la propia VM
sudo ip link set veth1 netns ue

sudo ip addr add 192.168.10.1/24 dev veth0
sudo ip link set veth0 up

sudo ip netns exec ue ip addr add 192.168.10.2/24 dev veth1
sudo ip netns exec ue ip link set veth1 up
sudo ip netns exec ue ip link set lo up


#Configurar enrutamiento
sudo ip route add 192.168.10.0/24 dev veth0 #Ruta hacia el namespace

sudo ip route add 10.60.0.1/32 via 192.168.10.1 #Ruta hacia la interfaz del namespace con N3IWF

sudo iptables -t nat -A PREROUTING -p udp --dport 1700 -j DNAT --to-destination 192.168.10.2:1700 #Reenvio datos a namespace

sudo iptables -A FORWARD -p udp -d 192.168.10.2 --dport 1700 -j ACCEPT #Aceptacion reenvio datos a namespace

sudo ip netns exec ue iptables -t nat -A PREROUTING -p udp --dport 1700 -j DNAT --to-destination 10.1.1.1:1700 #Reenvio a N3IWF

sudo ip netns exec ue iptables -t nat -A POSTROUTING -o tun3 -j MASQUERADE #Configuracion mascara cuando reenvian datos

sudo ip netns exec ue iptables -A FORWARD -p udp -d 10.1.1.1 --dport 1700 -j ACCEPT #Aceptacion reenvio datos a N3IWF
```

Figura 5.27: Configuración de enrutamiento de paquetes UDP en el cliente NWu

Capítulo 6

Pruebas de conectividad

En este capítulo, se exponen las pruebas que se han realizado para comprobar el correcto funcionamiento del diseño y la implementación para la integración de las redes LoRaWAN y 5G. Estas pruebas se realizan entre los distintos puntos de la red, con la finalidad de identificar posibles errores más fácilmente; y, finalmente, entre los puntos finales de la red.

6.1. Conectividad entre la mota y el *gateway*

En el desarrollo del proyecto, lo primero que se ha realizado es implementar y configurar la mota y el *gateway*. Es por eso, que la primera prueba consiste en probar que los mensajes enviados por la mota llegan correctamente al *gateway* vía LoRa.

Para esta prueba, primero se enciende la mota para que envíe mensajes como se muestra en la sección 5.2 y el *gateway* para que se conecte a la red Wi-Fi y espere a recibir mensajes. En la terminal de Visual Studio Code, se puede comprobar los mensaje que llegan al *gateway*. En la figura 6.1, se muestra como el *gateway* recibe los mensajes e información de los paquetes de datos, así como los que han sido reenviados:

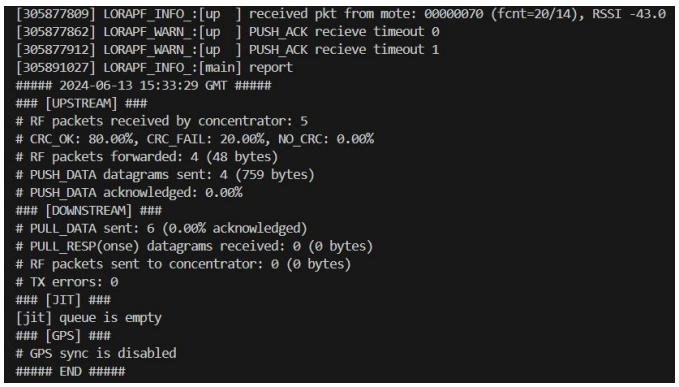


Figura 6.1: Paquetes de datos recibidos por el *gateway* que han sido enviados por la mota

6.2. Conectividad entre la mota y la Plataforma LoRaWAN

Una vez se comprobó la correcta conexión entre la mota y el *gateway*, se implementó y configuraron los servidores LoRaWAN usando la herramienta ChirpStack, como se explica en la sección 5.4. Después de tenerlos correctamente configurados, se comprobó que a la aplicación creada llegaban correctamente los mensajes enviados por la mota a través del *gateway*, que está conectado directamente a la plataforma LoRaWAN.

Para hacer esta comprobación, lo primero que se hizo fue configurar el reenvío de puertos en la máquina virtual donde se implementa la plataforma LoRaWAN, para que los paquetes UDP que lleguen al puerto 1700 del ordenador personal desde el *gateway*, se redirigan al puerto 1700 de la interfaz de la máquina virtual donde se implementa la plataforma LoRaWAN, como se mostró en la figura 5.25 que se hizo en el cliente NWu. Una vez hecho esto, se comprueba usando *tcpdump* para capturar los paquetes que llegan al puerto 1700. Es decir, muestra los paquetes de datos que llegan desde el gateway a la maquina virtual donde están implementados los servidores LoRaWAN.

```
20:37:11.377663 IP _gateway.49154 > juangui-VirtualBox.1700: RADIUS, Access-Accept (2), id: 0x01 length: 206
20:37:11.378887 IP juangui-VirtualBox.1700 > _gateway.49154:  [|radius]
20:37:11.696458 IP juangui-VirtualBox.1700 > _gateway.49155: RADIUS, Access-Accept (2), id: 0x5f length: 222
20:37:11.732020 IP _gateway.49155 > juangui-VirtualBox.1700:  [|radius]
20:37:14.322460 IP _gateway.49155 > juangui-VirtualBox.1700:  [|radius]
20:37:14.322827 IP juangui-VirtualBox.1700 > _gateway.49155:  [|radius]
20:37:19.332326 IP _gateway.49155 > juangui-VirtualBox.1700:  [|radius]
20:37:19.332595 IP juangui-VirtualBox.1700 > _gateway.49155:  [|radius]
20:37:24.343348 IP _gateway.49155 > juangui-VirtualBox.1700:  [|radius]
20:37:24.343531 IP juangui-VirtualBox.1700 > _gateway.49155:  [|radius]
20:37:29.352660 IP _gateway.49155 > juangui-VirtualBox.1700:  [|radius]
20:37:29.352975 IP juangui-VirtualBox.1700 > _gateway.49155:  [|radius]
20:37:30.537681 IP _gateway.49154 > juangui-VirtualBox.1700: RADIUS, Access-Accept (2), id: 0x01 length: 206
20:37:30.538738 IP juangui-VirtualBox.1700 > _gateway.49154:  [|radius]
20:37:30.855921 IP juangui-VirtualBox.1700 > _gateway.49155: RADIUS, Access-Accept (2), id: 0xd2 length: 222
```

Figura 6.2: Paquetes de datos que llegan desde el gateway a la maquina virtual donde están implementados los servidores LoRaWAN

Como se puede observar, los paquetes llegan correctamente al puerto 1700 de la máquina virtual desde el *gateway* cada 20 segundos aproximadamente. Después, se accede a la interfaz web. Desde ahí, se puede comprobar que el *gateway* y la mota están conectados al servidor como en la figura 5.11. En el apartado de *Gateways*, dentro de la organización creada, se puede acceder al *gateway* y ver las tramas LoRaWAN que recibe el servidor de red. Estas tramas aparecen encriptadas. En la figura 6.3 se puede ver el número de dispositivo, la banda de frecuencia, el CR y el SF, etcétera.



Figura 6.3: Paquetes de datos recibidos por el *gateway* desde la interfaz web de ChirpStack

Una vez comprobado que llegan correctamente al *gateway* y encriptados, se comprueba desde la aplicación creada, accediendo al dispositivo que se ha añadido. Desde ahí, se pueden ver los mensajes que llegan al servidor de aplicación, pero aquí desencriptadas (véase figura 6.4):

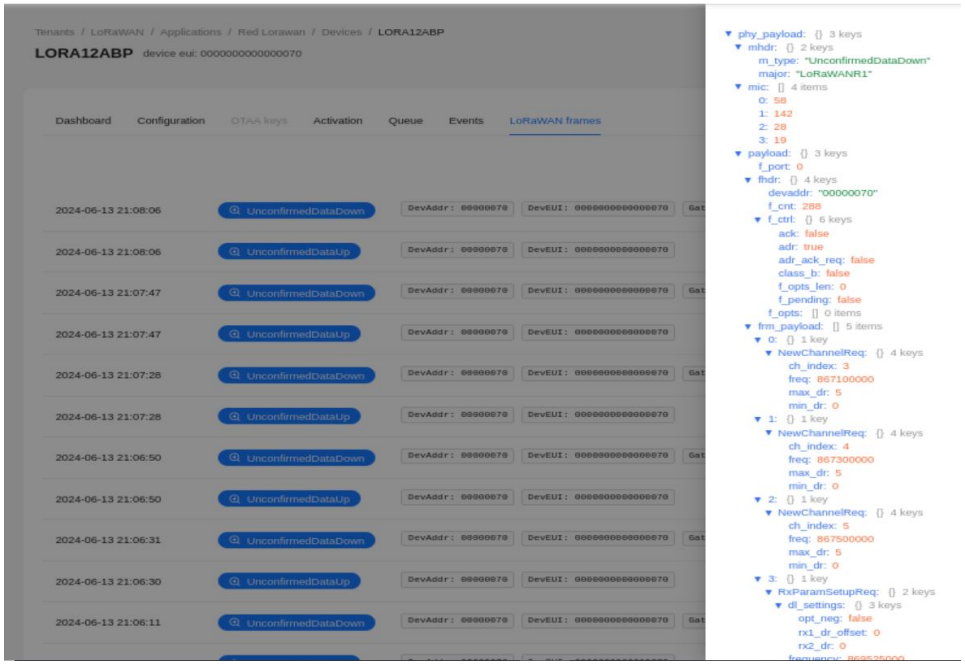


Figura 6.4: Paquetes de datos recibidos por el el servidor LoRaWAN, vistos desde la aplicación creada en ChirpStack

Como se puede comprobar, los paquetes de datos enviados por la mota llegan correctamente al servidorLoRaWAN, a través del *gateway*.

6.3. Conectividad entre el cliente NWu y el *Core* 5G

El objetivo de realizar esta prueba, es comprobar que la conexión entre el cliente NWu y el N3IWF se hace correctamente, siguiendo el proceso de autenticación IKEv2, donde se crea el túnel IPSec correctamente entre ellos. Para la realización de la prueba, en primer lugar, se lanza el *core* 5G, como se ha explicado en la sección 5.5:

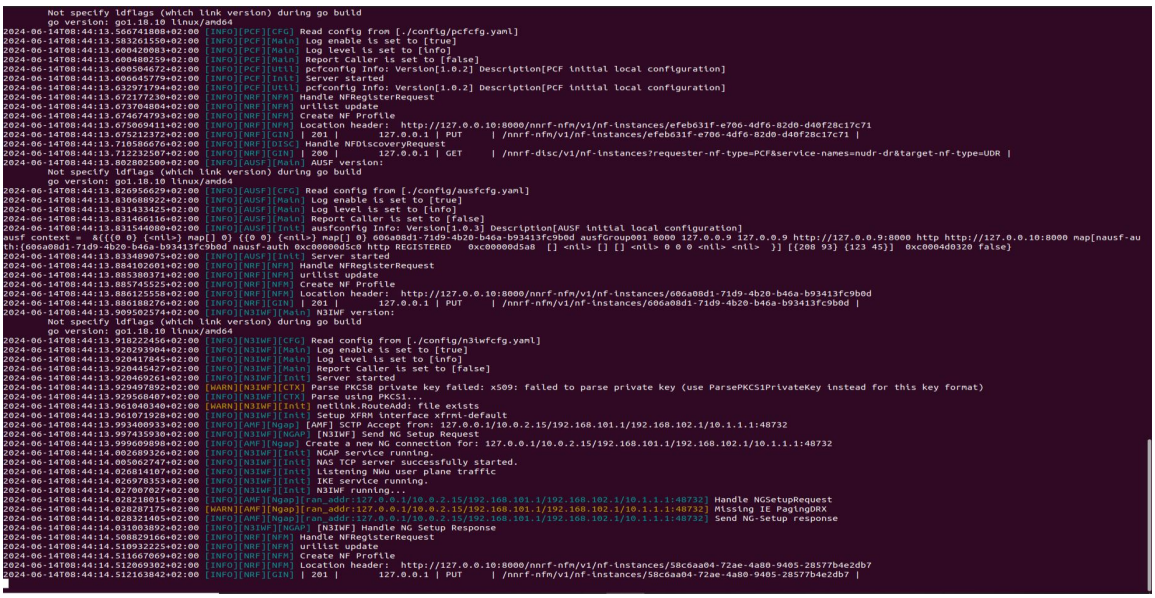


Figura 6.5: Lanzamiento del servidor del *core* 5G en funcionamiento usando free5gc

Una vez está funcionando el servidor del *core* 5G, se lanza el cliente NWu. Al lanzarlo, se observan los estados que sigue la autenticación y la creación del túnel IPSec con el *core* 5G, a través del N3IWF. En el estado 1, se hace inicio del proceso de autenticación IKEv2, en el que se negocián los parámentros criptográficos, las claves de autenticación se van a intercambiar mediante el protocolo *Diffie-Helmann* (recuadro rojo). Finalmente, se comprueba que las claves que tiene el cliente NWu y el N3IWF son correctas (recuadro amarillo). Este proceso se muestra en la figura 6.6:

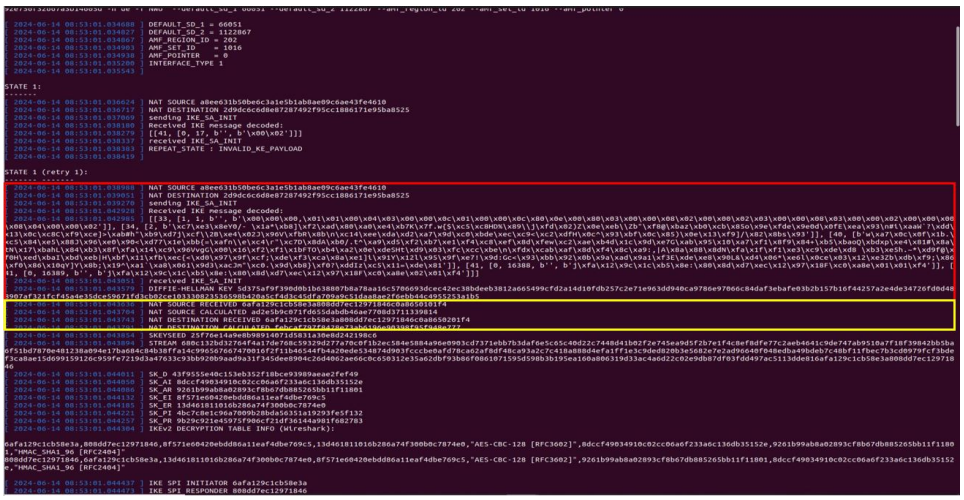


Figura 6.6: Conexión entre el cliente NWu y el *core* 5G (estado 1)

En el estado 2, el cliente NWu envía un mensaje IKE de autenticación, que contiene un mensaje *Extensible Authentication Protocol* EAP (recuadro rojo). Después, recibe una respuesta a este mensaje del 5GC, la decodifica y comprueba la correcta autenticación del dispositivo y los certificados (recuadro amarillo). Es decir, verifica la identidad del dispositivo.

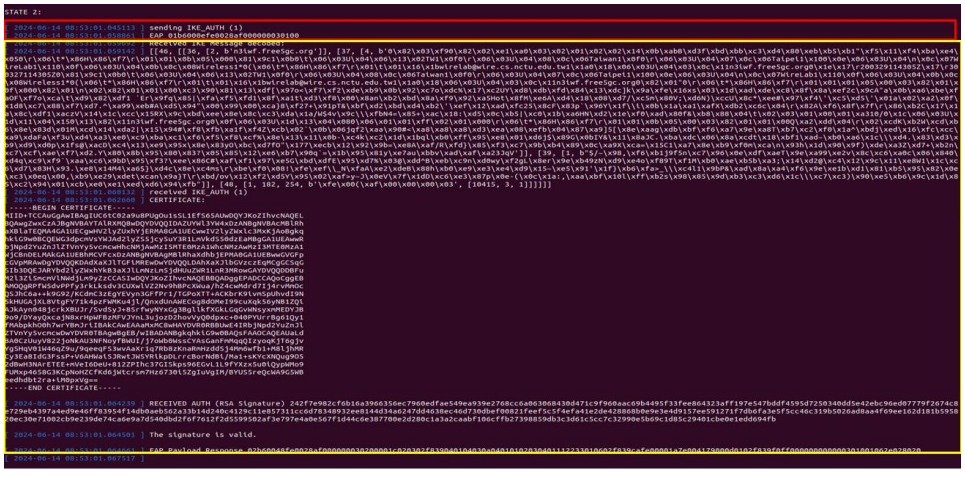


Figura 6.7: Conexión entre el cliente NWu y el *core* 5G (estado 2)

En el estado 3, se envía otro mensaje IKE para autenticarse con los elementos del 5GC (recuadro rojo), y se recibe otro mensaje de respuesta NAS con las claves de autenticación de los distintos elementos del 5GC (recuadro amarillo).

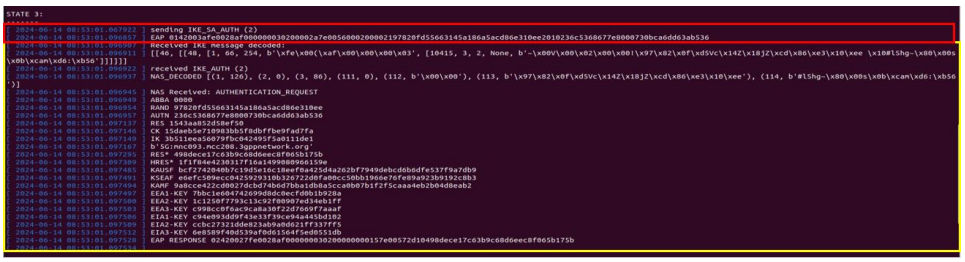


Figura 6.8: Conexión entre el cliente NWu y el *core* 5G (estado 3)

En el estado 4, se envía otro mensaje IKE de autenticación al 5GC (recuadro rojo), y se recibe otro mensaje de respuesta que contiene datos NAS (recuadro amarillo). Se comprueba que el mensaje no ha sido modificado durante la comuniación mediante el cálculo de MAC (recuadro verde) y se reciben las claves para comunicarse con el N3IWF (recuadro blanco).



Figura 6.9: Conexión entre el cliente NWu y el *core* 5G (estado 4)

En el estado 5, se envía otro mensaje IKE al 5GC (recuadro rojo), y se recibe otra mensaje de respuesta para finalizar la autenticación (recuadro amarillo).



Figura 6.10: Conexión entre el cliente NWu y el *core* 5G (estado 5)

En el estado 6, se establecen los parámetros para establecer el túnel para el envío de datos TCP y para establecer el túnel IPSec mediante el intercambio de las direcciones IP (recuadro rojo). Después de este envío de mensajes, mediante la señalización SA Child, se establece el túnel para la sesión TCP (recuadro amarillo).

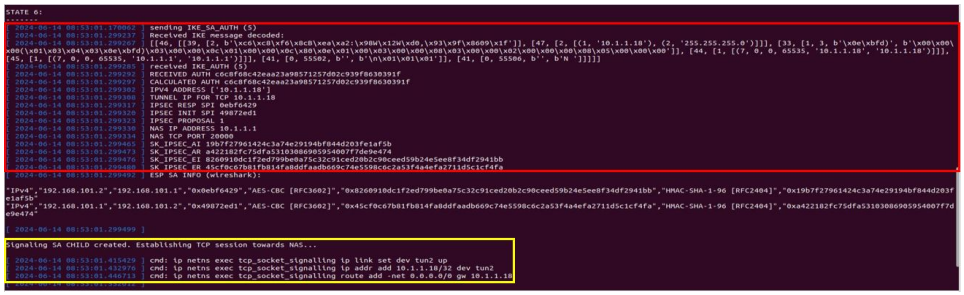


Figura 6.11: Conexión entre el cliente NWu y el *core* 5G (estado 6)

Finalmente, en el estado 7, se completa el registro del cliente NWu en el *core* 5G (recuadro rojo). Además, se intercambian las claves IPSec (recuadro amarillo) y se crea el túnel para el envío de paquetes de datos UDP entre el cliente NWu y el N3IWF (recuadro verde).

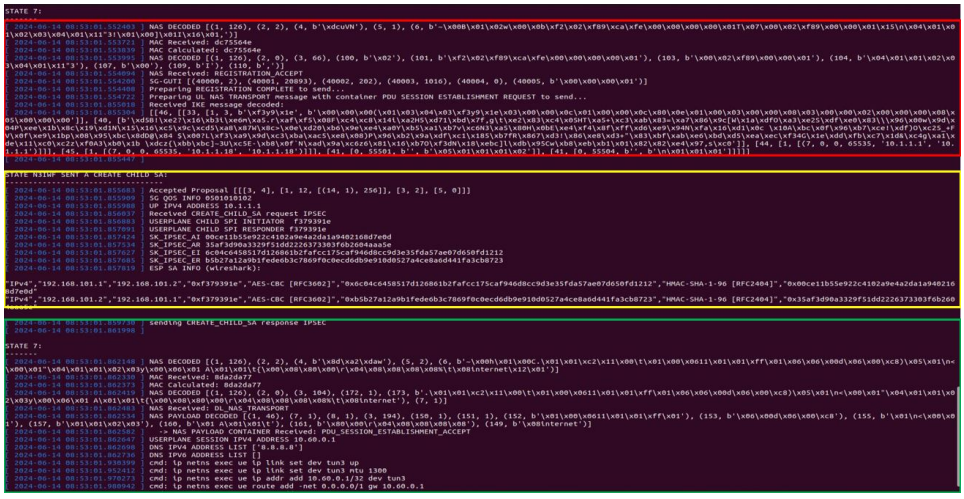


Figura 6.12: Conexión entre el cliente NWu y el *core* 5G (estado 7)

Como se puede comprobar en la figura 6.12, se realiza correctamente todo el proceso de conexión, autenticación y creación del túnel entre el cliente NWu y el N3IWF.

6.4. Conectividad entre la mota y la Plataforma LoRaWAN a través del *Core* 5G, sin que participe el cliente NWu

Con la realización de esta prueba, se quiere comprobar que se puede enrutar el tráfico que entra al puerto 1700 del *core* 5G hacia el servidor de

LoRaWAN correctamente. Para comprobar esto, los primero que se realiza es encender la mota para que envíe mensajes y el *gateway*. Acto seguido, se configura el reenvío de puertos en la máquina virtual donde se implementa el *core* 5G como en la figura 5.25, y se comprueba usando *tcpdump* que llegan correctamente a la máquina, se va a realizar el seguimiento de la trama enviada a las 13:22:05:

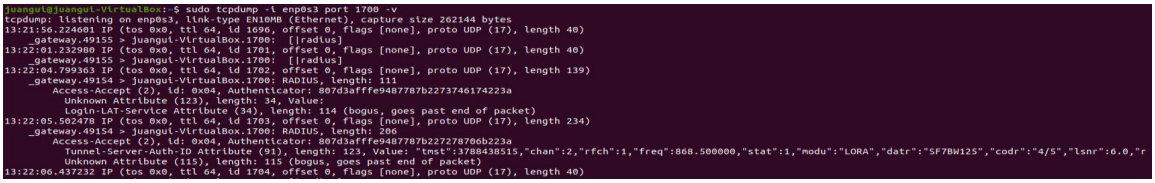


Figura 6.13: Comprobación con *tcpdump* de que llegan los mensajes de la mota a la máquina del *core* 5G

Una vez se ha comprobado esto, se arranca la plataforma de LoRaWAN en su máquina virtual y el *core* 5G. Y, se configura el enrutamiento con el comando explicado en la sección 5.5:

```
sudo iptables -t nat A PREROUTING -p udp --dport 1700 -j DNAT
    --to-destination 192.168.102.2:1700
```

Se comprueba, en la figura 6.14, la configuración de *iptables* para ver que se ha configurado correctamente:

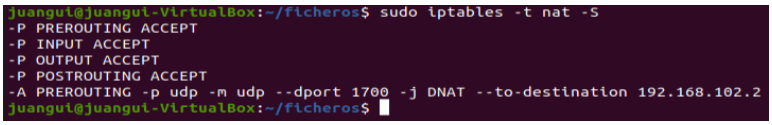


Figura 6.14: Comprobación de *iptables* en la máquina virtual donde se implementa el *core* 5G

Ahora, se comprueba en *Wireshark* que los mensajes que llegan al puerto 1700 se están reenviando a la máquina de la plataforma LoRaWAN:

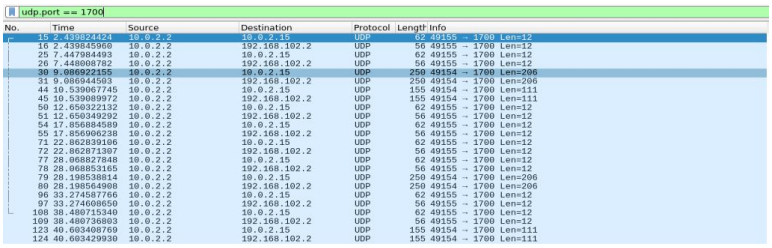


Figura 6.15: Comprobación de reenvío de mensajes en *Wireshark* del *core* 5G al servidor LoRaWAN, en la máquina virtual del *core* 5G

Y, se realiza la misma comprobación en la máquina dela plataforma LoRaWAN. Se puede observar como le llegan los mensajes y los reenvía al *gateway bridge*, con IP 172.18.0.4. A su vez, se puede observar el mensaje de respuesta de vuelta:

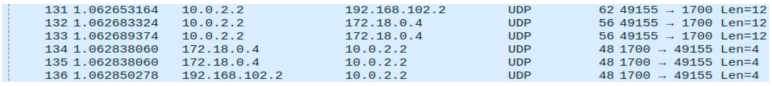


Figura 6.16: Comprobación de reenvío de mensajes en *Wireshark* del *core* 5G al servidor LoRaWAN, en la máquina virtual del servidor LoRaWAN

Después de hacer estas comprobaciones, en la interfaz web se puede ver como se ha detectado al dispositivo y al *gateway*. Del mismo modo, se pueden observar los mensajes que llegan al *gateway*, que aparecen encriptados, y, los que llegan a la aplicación, que aparecen sin encriptar. Esto se muestra en las figuras 6.17, 6.18 y 6.19:

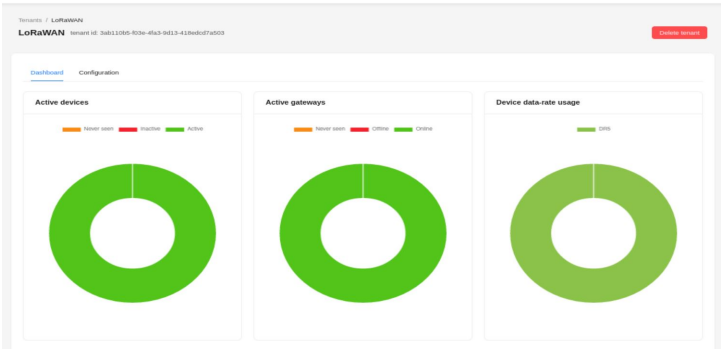


Figura 6.17: Comprobación de detección de dispositivos en ChirpStack

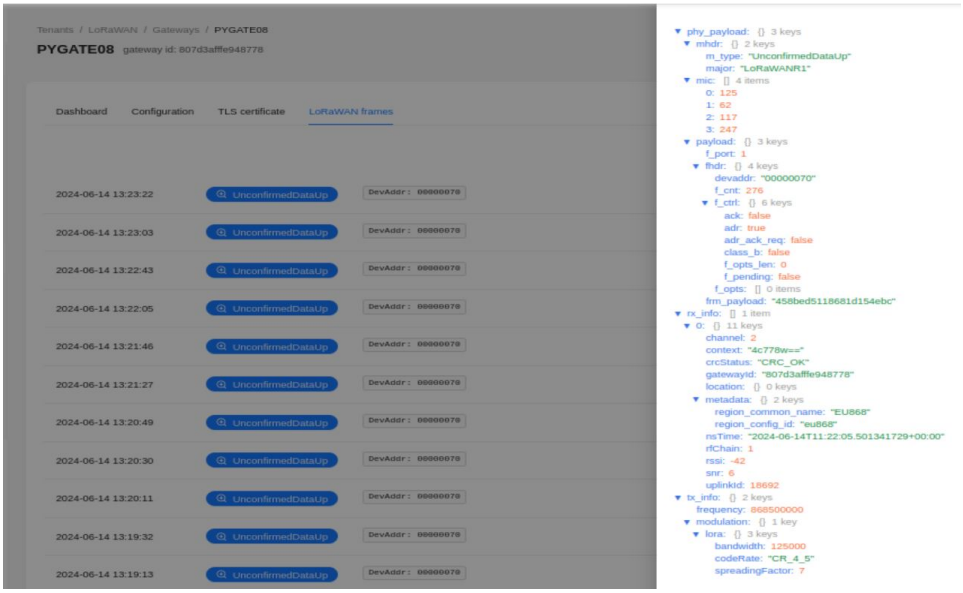


Figura 6.18: Comprobación de mensaje recibido por el *gateway* desde ChirpStack

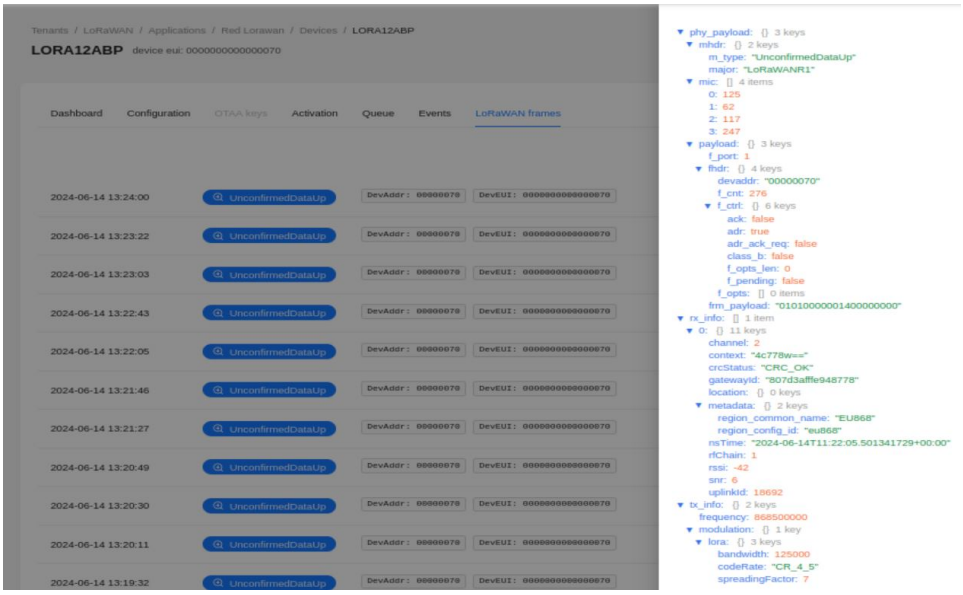


Figura 6.19: Comprobación de mensaje recibido por el servidor LoRaWAN a través del *core* 5G desde la aplicación de ChirpStack

Como se puede comprobar, los mensajes enviados por la mota llegan correctamente a la plataforma LoRaWAN a través del *core* 5G.

6.5. Conectividad End-to-End (E2E)

El objetivo de esta prueba, es comprobar que la red diseñada y mostrada en la figura 5.1 funciona correctamente tras la implementación y configuración de sus elementos. Es decir, comprobar que se puede hacer la integración de las redes LoRaWAN y 5G.

Para realizar esta prueba, lo primero que se realiza es encender la mota y el *gateway*, comprobar que al cliente NWu le llegan los mensajes enviados por la mota a través del *gateway*, haciendo uso del reenvío de puertos configurado en la figura 5.25. Se va hacer el seguimineto del mensaje enviado a las 18:45:23 por la mota. En la figura 6.20, se puede comprobar como llega a la máquina virtual del cliente NWu.

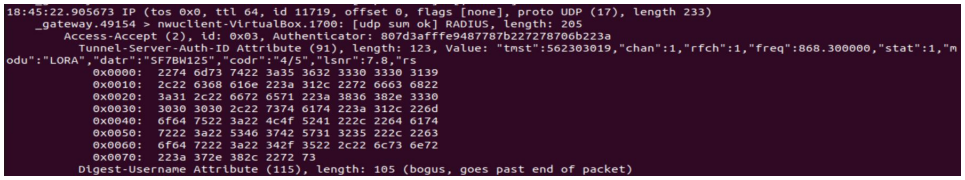


Figura 6.20: Comprobación del mensaje de la mota recibido por la máquina virtual donde se implementa el cliente NWu

Una vez comprobado esto, se arranca el servidor LoRaWAN, el *core* 5G y el cliente NWu. Como se ha demostrado en la prueba 6.3, el cliente NWu y el *core* 5G realizan correctamente el proceso de conexión, autenticación y creación del túnel entre el cliente NWu y el N3IWF. Después de esto, se configura el enrutamiento en la máquina virtual donde se implementa el *core* 5G, como en la prueba 6.4, esta configuración se muestra en la figura 6.14. Al mismo tiempo, se configura el enrutamiento en la máquina virtual donde se implementa el cliente NWu, como se explica en la figura 5.27 en la sección 5.6. En la figura 6.21, se muestra que la configuración se realiza correctamente, tanto en la propia máquina virtual, como en el *namespace* que se crea al conectarse el cliente NWu y el N3IWF:

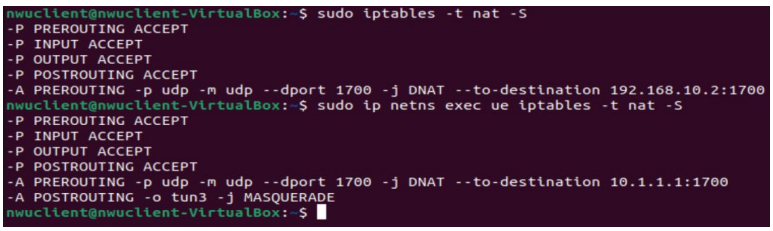


Figura 6.21: Comprobación de *iptables* en la máquina virtual donde se implementa el cliente NWu

Cuando se ha comprobado que las configuraciones de enrutamiento en todas las partes que componen el diseño de la red son correctas, se comprueba que los paquetes llegan correctamente al *namespace*, de la máquina del cliente NWu; y, los envía al N3IWF a través del túnel IPSec que se crea entre ellos. Esta comprobación, se realiza haciendo uso de *tcpdump*:

```
18:45:22.905746 IP (tos 0x0, ttl 62, id 11719, offset 0, flags [none], proto UDP (17), length 233)
    10.60.0.1.49154 > 10.1.1.1.1700: RADIUS, length: 205
        Access-Accept (2), id: 0x03, Authenticator: 807d3afffe9487787b227278706b223a
          Tunnel-Server-Auth-ID Attribute (91), length: 123, Value: "tmst":562303019,"chan":1,"rfch":1,"freq":868.300000,"stat":1,"m
odu":"LORA","datr":"SF7BW125","codr":"4/5","lsnr":7.8,"rs
          Digest-Username Attribute (115), length: 105 (bogus, goes past end of packet)
```

Figura 6.22: Comprobación del mensaje recibido por el *namespace* de la máquina del cliente NWu

Ahora, se comprueba que llega correctamente al N3IWF y lo envía al UPF a través del túnel GTP que se crea entre ellos. Acto seguido, el UPF lo envía a la plataforma LoRaWAN. Esta comprobación se realiza usando *Wireshark* en la máquina virtual para poder observar mejor como se reenvía el mensaje de un elemento a otro.

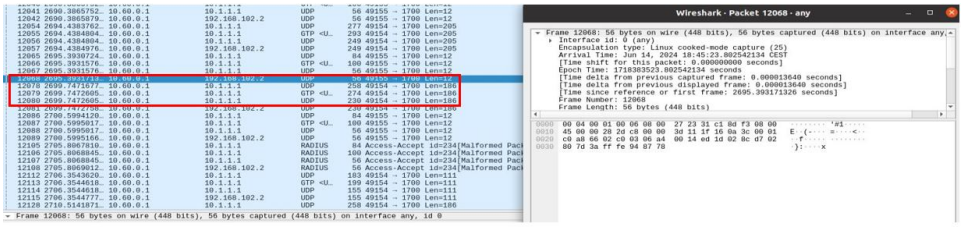


Figura 6.23: Comprobación del reenvío del mensaje recibido por el N3IWF hacia el servidor LoRaWAN

El siguiente paso, es comprobar que llegan los paquetes correctamente a la plataforma LoRaWAN y los envía a la apliación a través del *gateway bridge*; y, que le envía el mensaje de respuesta de vuelta. Al igual que la anterior comprobación, esta también se va a realizar con *Wireshark* para poder observar como se reenvía el mensaje de un elemento a otro.

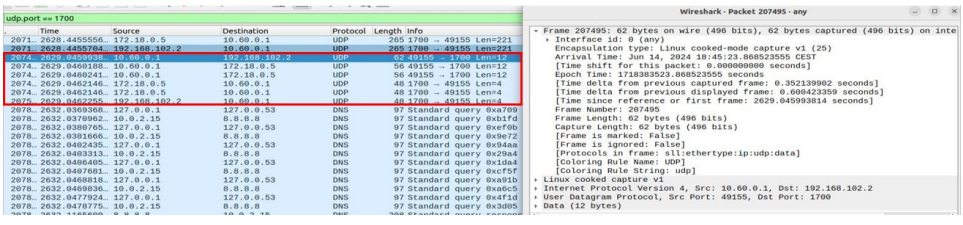


Figura 6.24: Comprobación del reenvío del mensaje recibido por el servidor LoRaWAN

Por último, se comprueba que en la interfaz web de ChirpStack se puede

ver como se ha detectado al dispositivo y al *gateway*. Además, se pueden observar los mensajes que llegan al *gateway*, que aparecen encriptados, y, los que llegan a la aplicación, que aparecen desencriptados. Al igual que en la prueba 6.4. Esto se muestra en las figuras 6.25, 6.26 y 6.27:

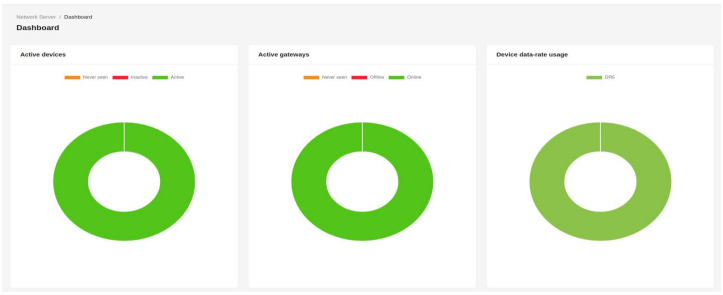


Figura 6.25: Comprobación de detección de dispositivos en ChirpStack, prueba entre puntos finales

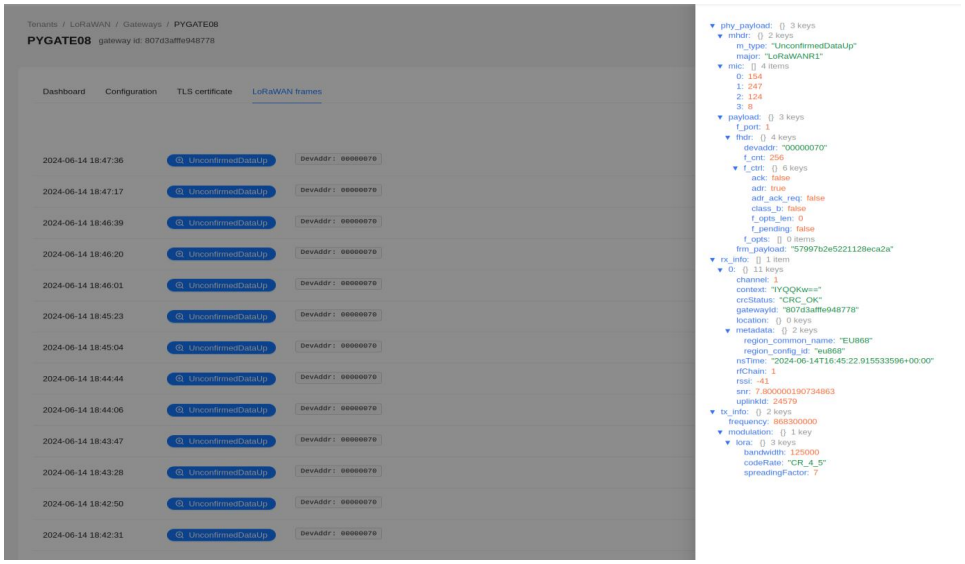


Figura 6.26: Comprobación de mensaje recibido por el *gateway* desde ChirpStack, prueba entre puntos finales

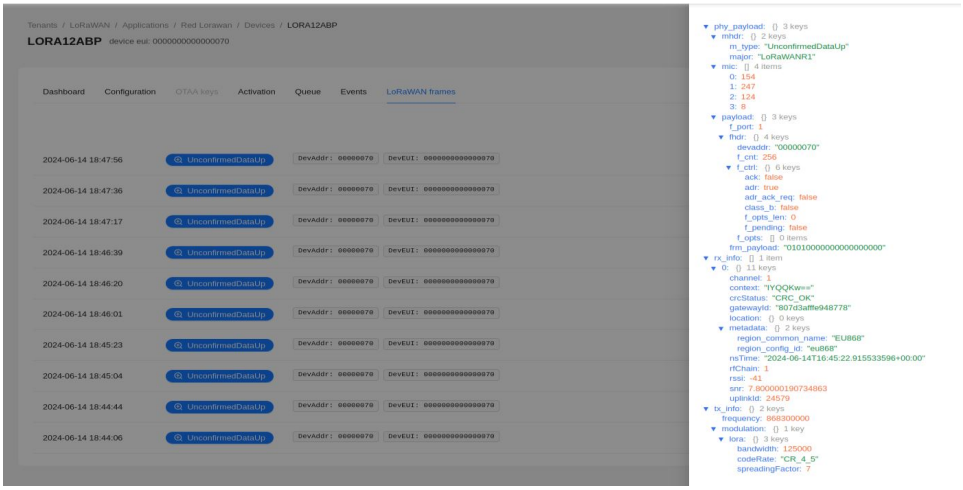


Figura 6.27: Comprobación de mensaje recibido por el servidor LoRaWAN desde la aplicación de ChirpStack, prueba entre puntos finales

Finalmente, se comprueba que los mensajes llegan correctamente desde la mota hasta el servidor LoRaWAN a través del *core* 5G y haciendo uso del cliente NWu. A partir de esta prueba, se puede concluir que la integración entre ambas redes es posible de realizar y funciona de manera correcta.

Capítulo 7

Conclusiones y líneas futuras

En este capítulo, se exponen las conclusiones que se han obtenido durante la realización del proyecto y se plantean posibles líneas de trabajo futuras.

7.1. Conclusiones

Este proyecto, ha supuesto algunas dificultades durante la instalación de los módulos o paquetes necesarios para el uso de las herramientas utilizadas. Estas dificultades surgieron debido a incompatibilidades entre el ordenador personal en el que se realizó el proyecto y algunos de estos paquetes.

El objetivo principal de este proyecto, era la integración de las red LoRaWAN y 5G. Para ello, en primer lugar, se estudiaron las principales características teóricas de ambas redes, los elementos que las componen y su funcionamiento. Al mismo tiempo, se estudiaron diversos proyectos realizados en la Universidad de Granada relacionados con LoRaWAN y que podían ser de ayuda para la realización de este. Se analizaron las distintas herramientas que existen actualmente y que se podían utilizar para llevar a cabo el proyecto. En consecuencia, se eligieron las herramientas más adecuadas para lograr los objetivos propuestos en el proyecto, como se explica en el capítulo 2.

Una vez se hiceron estos estudios, se diseñó la arquitectura de la red para la integración de la red LoRaWAN y la red 5G. Los elementos necesarios para ello: mota, *gateway*, cliente NWu y los servidores del *core* 5G y de LoRaWAN. Acto seguido, se hicieron las implementaciones y configuraciones de cada uno de los elementos que forman la red, como se explica en el capítulo 5. Cabe mencionar, que se encontraron dificultades a la hora de la instalación de free5gc, por incompatibilidades de la versión de MongoDB que utilizaba. Esto se resolvió utilizando una versión distinta del software de Ubuntu en la máquina donde se implementa el *core* 5G que es compatible

con versiones anteriores de MongoDB y que si soportaba la virtualización del ordenador personal. La configuración de los distintos elementos de la red, se llevó a cabo trabajando con distintos lenguajes de programación, como C++ o MicroPython, y aplicaciones software.

Después de configurar e implementar los distintos elementos, se analizó como llevar a cabo el enrutamiento de los mensajes entre los distintos elementos y se hicieron pruebas de conectividad entre ellos, como se muestra en el capítulo 6. Primero, se hicieron pruebas entre elementos y, luego, entre los dispositivos finales, con el objetivo de localizar errores más fácilmente y corregirlos.

Como se ha demostrado, la red diseñada funciona correctamente y, es posible realizar la integración de la redes LoRaWAN y 5G. Por tanto, se considera que se han alcanzado todos los objetivos planteados en este proyecto.

7.2. Líneas futuras

A partir de la red diseña, la cual funciona al 100 %, se podrían proponer mejoras o líneas de desarrollo adicionales como:

- Implementación de creación de distintos túneles usando *Network Slice Selection* (NSS) para enrutar los mensajes hacia distintos puntos finales o aplicaciones.
- Implementación del envío de mensajes a través de apliaciones de mensajería instantanea entre dispositivos finales.
- Implementación y despliegue de la red diseñada con equipamiento real para hacer pruebas de campo. Por ejemplo, en entornos rurales o industriales.
- Analizar la capacidad, la tasa de datos y el consumo de la red, así como tratar de optimizar los resultados preliminares.
- Implementación de *Global Positioning System* (GPS) para la localización de los dispositivos finales, es decir, las motas LoRaWAN.
- Diseño e implementación de una web que permita motorizar, ajustar y controlar los dispositivos que forma la red diseñada.

Bibliografía

[1] Yasser Ismail. "Introductory chapter: Internet of things (iot) importance and its applications". En: (2019).

[2] Maciej Piechowiak, Piotr Zwierzykowski y Bartosz Musznicki. "LoRaWAN Metering Infrastructure Planning in Smart Cities". En: *Applied Sciences* 13 (2023).

[3] Quy Vu Khanh et al. "Wireless communication technologies for IoT in 5G: Vision, applications, and challenges". En: *Wireless Communications and Mobile Computing* 2022 (2022), págs. 1-12.

[4] Naser Al-Falahy y Omar Y. Alani. "Technologies for 5G Networks: Challenges and Opportunities". En: *IT Professional* 19.1 (2017), págs. 12-20. DOI: 10.1109/MITP.2017.9.

[5] Imen Bouazzi et al. "Future trends for healthcare monitoring system in smart cities using LoRaWAN-based WBAN". En: *Mobile Information Systems* 2022 (2022).

[6] Badreddine Miles et al. "A study of LoRaWAN protocol performance for IoT applications in smart agriculture". En: *Computer Communications* 164 (2020), págs. 148-157.

[7] Chirpstack. *The ChirpStack project*. URL: https://www.chirpstack.io/docs/index.html (visitado 10-05-2024).

[8] ChirpStack. *ChirpStack Architecture*. URL: https://www.chirpstack.io/docs/architecture.html (visitado 20-05-2024).

[9] The Things Network. *The Things Network Official Web Page*. URL: https://www.thethingsnetwork.org/ (visitado 06-06-2024).

[10] The Things Network. *The Things Stack Architecture*. URL: https://www.thethingsindustries.com/docs/the-things-stack/concepts/architecture/ (visitado 06-06-2024).

[11] Loriot. *Loriot Official Web Page*. URL: https://loriot.io/ (visitado 06-06-2024).

[12] Mioty Alliance. *Mioty Alliance Official Web Page*. URL: https://mioty-alliance.com/ (visitado 06-06-2024).

[13] Loriot. *Loriot LoRaWAN Network Server*. URL: https://loriot.io/lorawan-network-server.html (visitado 06-06-2024).

[14] Loriot. *Loriot Mioty Service Server*. URL: https://loriot.io/mioty-service-center.html (visitado 06-06-2024).

[15] Open5gs. *Open5gs*. URL: https://open5gs.org/ (visitado 19-05-2024).

[16] Open5gs. *Quickstart Open5gs*. URL: https://open5gs.org/open5gs/docs/guide/01-quickstart/ (visitado 19-05-2024).

[17] free5GC. *free5GC*. URL: https://free5gc.org/ (visitado 19-05-2024).

[18] free5GC. *free5GC repository*. URL: https://github.com/free5gc/free5gc (visitado 19-05-2024).

[19] OpenAirInterface. *About OpenAirInterface Source Code*. URL: https://openairinterface.org/overview/ (visitado 07-06-2024).

[20] OpenAirInterface. *OpenAirInterface RAN Project Group*. URL: https://openairinterface.org/oai-5g-ran-project/ (visitado 07-06-2024).

[21] OpenAirInterface. *OpenAirInterface 5G Core Network*. URL: https://openairinterface.org/oai-5g-core-network-project/ (visitado 07-06-2024).

[22] Fabricio Ferraz. *NWu-Non3GPP-5GC client*. URL: https://github.com/fasferraz/NWu-Non3GPP-5GC (visitado 18-05-2024).

[23] Ali Güngör. *UERANSIM*. URL: https://github.com/aligungr/UERANSIM (visitado 19-05-2024).

[24] Software Radio Systems. *srsRAN Project*. URL: https://www.srsran.com/ (visitado 07-06-2024).

[25] Software Radio Systems. *srsRAN Software Architecture*. URL: https://docs.srsran.com/projects/project/en/latest/dev_guide/source/software_arch/source/index.html (visitado 07-06-2024).

[26] Jorge Navarro-Ortiz et al. "Integration of LoRaWAN and 4G/5G for the Industrial Internet of Things". En: *IEEE Communications Magazine* 56.2 (2018), págs. 60-67. DOI: 10.1109/MCOM.2018.1700625.

[27] Felix Delgado-Ferro et al. "A LoRaWAN Architecture for Communications in Areas without Coverage: Design and Pilot Trials". En: *Electronics* (). DOI: 10.3390/electronics11050804. URL: https://www.mdpi.com/2079-9292/11/5/804.

[28] Ala' Khalifeh et al. "A Survey of 5G Emerging Wireless Technologies Featuring LoRaWAN, Sigfox, NB-IoT and LTE-M". En: (2019), págs. 561-566. DOI: 10.1109/WiSPNET45539.2019.9032817.

[29] LoRa Alliance. "LoRaWAN - What is? A technical overview of LoRa and LoRaWAN." En: (2015).

[30] Sylvain Montagny. "LoRa-LoRaWAN and Internet of Things." En: (2022).

[31] Sébastien Maudet et al. "Refined Node Energy Consumption Modeling in a LoRaWAN Network". En: *Sensors* (2021).

[32] Shilpa Devalal y A. Karthikeyan. "LoRa Technology - An Overview". En: (2018), págs. 284-290.

[33] Ankita Raychowdhury y Ankita Pramanik. "Survey on LoRa technology: solution for internet of things". En: *Intelligent Systems, Technologies and Applications: Proceedings of Fifth ISTA 2019, India* (2020), págs. 259-271.

[34] Ahmad Nazhan Rosli et al. "Implementation of MQTT and LoRaWAN System for Real-time Environmental Monitoring Application". En: (2020), págs. 287-291.

[35] Dipa Soni y Ashwin Makwana. "A survey on mqtt: a protocol of internet of things (iot)". En: 20 (2017), págs. 173-177.

[36] Valerie Lampkin et al. "Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry". En: (2012).

[37] Alon Ronder. *The 4-way handshake WPA/WPA2 encryption protocol.* URL: `https://medium.com/@alonr110/the-4-way-handshake-wpa-wpa2-encryption-protocol-65779a315a64` (visitado 28-05-2024).

[38] Rupendra Nath Mitra y Dharma P Agrawal. "5G mobile technology: A survey". En: *ICT express* 1.3 (2015), págs. 132-137.

[39] Jorge Navarro-Ortiz et al. "A Survey on 5G Usage Scenarios and Traffic Models". En: *IEEE Communications Surveys Tutorials* 22.2 (2020), págs. 905-929. DOI: `10.1109/COMST.2020.2971781`.

[40] Marek Średniawa y Dariusz Bursztynowski. "Apuntes tema 1 asignatura Next Generation Networks". En: (2022).

[41] 3GPP. *The Third Generation Partnership Project (3GPP).* URL: `https://www.3gpp.org/about-us` (visitado 08-05-2024).

[42] ETSI. "System architecture for the 5G System (5GS)". En: (2021).

[43] ETSI. "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; 5G; Release description; Release 15". En: (2019).

[44] Windows. *Cómo buscar las especificaciones del equipo y los requisitos del sistema para Windows 10.* URL: `https://www.microsoft.com/es-es/windows/windows-10-specifications` (visitado 10-05-2024).

[45] Arduino. *Downloads Arduino IDE.* URL: `https://www.arduino.cc/en/software` (visitado 10-05-2024).

[46] Visual Studio. *Visual Studio Code*. URL: https://code.visualstudio.com/ (visitado 10-05-2024).

[47] VirtualBox. *Welcome to VirtualBox.org*. URL: https://www.virtualbox.org/ (visitado 10-05-2024).

[48] Ubuntu. *Download Ubuntu*. URL: https://ubuntu.com/download (visitado 10-05-2024).

[49] Docker. *Docker Compose overview*. URL: https://docs.docker.com/compose/ (visitado 18-05-2024).

[50] Wireshark. *Download Wireshark*. URL: https://www.wireshark.org/download.html (visitado 10-05-2024).

[51] Overleaf. *Overleaf*. URL: https://es.overleaf.com/project (visitado 10-05-2024).

[52] GanttProject. *Download GanttProject 3.3*. URL: https://www.ganttproject.biz/download/free (visitado 04-06-2024).

[53] Lenovo. *ThinkPad E15 Gen 2 (Intel)*. URL: https://lenovo.com/es/es/p/laptops/thinkpad/thinkpade/e15-g2/22tpe15e5n2?orgRef=https%253A%252F%252Fwww.google.com%252Ft (visitado 16-05-2024).

[54] Pycom Invent. *Pygate*. URL: https://docs.pycom.io/tutorials/expansionboards/pygate/ (visitado 16-05-2024).

[55] Jorge Navarro Ortiz. *Repositorio en GitHub "testbed-LoRaWAN"*. URL: https://github.com/jorgenavarroortiz/testbed-LoRaWAN/tree/main (visitado 08-06-2024).

[56] ChirpStack. *Quickstart Docker Compose*. URL: https://www.chirpstack.io/docs/getting-started/docker.html (visitado 09-06-2024).

[57] free5gc. *Free5gc Installation*. URL: https://free5gc.org/guide/3-install-free5gc/ (visitado 11-06-2024).

[58] Digital Ocean. *How To Install MongoDB on Ubuntu 20.04*. URL: https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-20-04 (visitado 11-06-2024).

[59] Jorge Navarro Ortiz. *Repositorio en GitHub "6GLoRaGRAN-connectivity"*. URL: https://github.com/jorgenavarroortiz/6GLoRaGRAN-connectivity/tree/main/free5gc (visitado 11-06-2024).

[60] Mitshell. *Card*. URL: https://github.com/mitshell/card (visitado 12-06-2024).

[61] Mitshell. *CryptoMobile*. URL: https://github.com/mitshell/CryptoMobile (visitado 12-06-2024).

Apéndice A

Manual de Usuario

En este apéndice, se explica como usar el proyecto realizado sobre integración de redes LoRaWAN y 5G, para que pueda ser utilizado para futuros desarrollos a partir de él y pruebas.

En primer lugar, hay que encender las motas y el *gateway*, asegurándose de que el *gateway* está bien configurado con la red de Internet, como se explica en 5.3. Una vez hecho esto, se enciende la máquina virtual donde se implementa el servidor LoRaWAN y se arranca el servidor de LoRaWAN, introduciendo los siguiente comandos:

```
cd chirpstack-docker
sudo docker-compose -d
```

Cuando ya está arrancado el servidor, se accede a la interfaz web de ChirpStack en la dirección *http://localhost:8080* y se introduce el usuario *admin* y la contraseña *admin*. Lo siguiente es realizar la configuración de la red, con los dispositivos LoRaWAN que forman parte de ella. A continuación, se explica como:

1. **Creación de una organización**: en el apartado *tenant* de la interfaz web se crea una organización. En ella, se incluirán los distintos *gateways* y aplicaciones.

2. **Creación de *gateways***: dentro de la organización creada, en el apartado *Gateways*, se añaden las pasarelas que componen la red. Se añade el nombre con el que se quiere identificar a cada uno, y su identificador de dispositivo en *Gateway ID*.

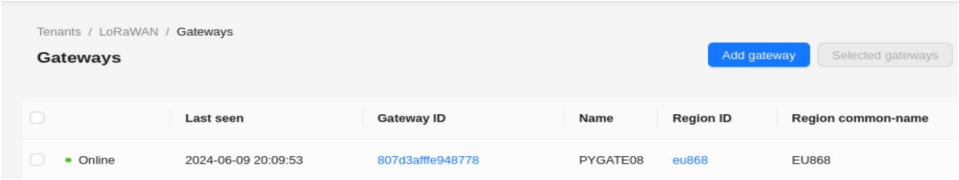


Figura A.1: Creación de *gateways* en ChirpStack

3. **Creación de perfil de dispositivo**: en la sección *Device Profiles*, dentro de la organización creada, se crean los perfiles de los dispositivos. Cuando se crea un perfil de dispositivo, hay que añadir el nombre con el que se va a identificar, la región y frecuencia en la que está, en este caso EU868; la versión MAC del dispositivo, el algoritmo ADR, en este caso el que viene por defecto (solo de LoRa) y, añadiendo el intervalo de tiempo máximo esperado de *uplink* en segundos. En este caso, la activación es mediante ABP, por lo que no se configura la activación mediante OTAA en el dispositivo.

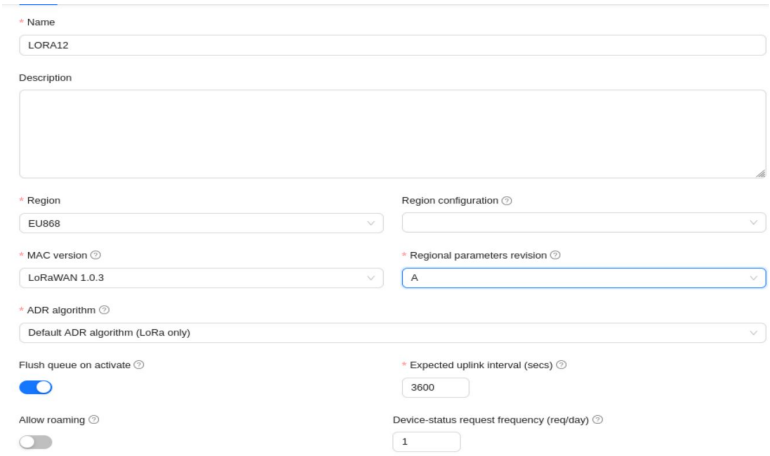


Figura A.2: Creación de perfil de dispositivo en ChirpStack

4. **Creación de aplicación**: también dentro de la organización creada, se crea la aplicación, en el apartado *Applications*, con el nombre con el que se quiera identificar. Dentro de la aplicación creada, se añaden los equipos, indicando su nombre, el *device Extended Unique Identifier 64 bits* (EUI64) y el *join* EUI64, se elige el perfil del usuario de ese dispositivo y, se deshabilita la validación por contador de trama. Además, en el apartado de *Activation* del equipo añadido, se indica su dirección y su NwkSKey y AppSKey. En las figuras A.3 y A.4 se muestra la configuración del dispositivo dentro de la aplicación.

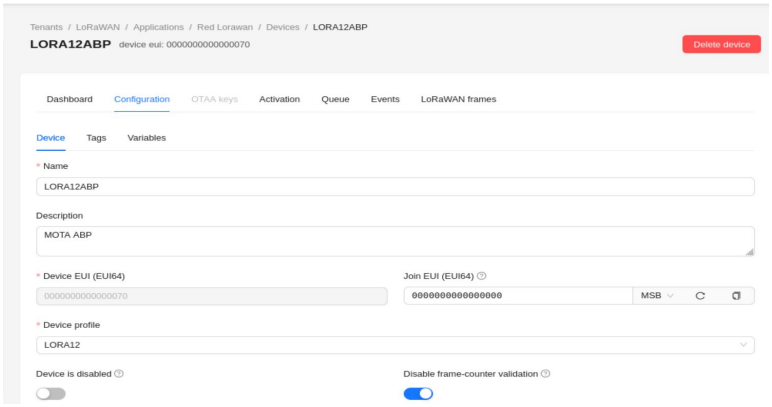


Figura A.3: Configuración de dispositivo en aplicación de ChirpStack

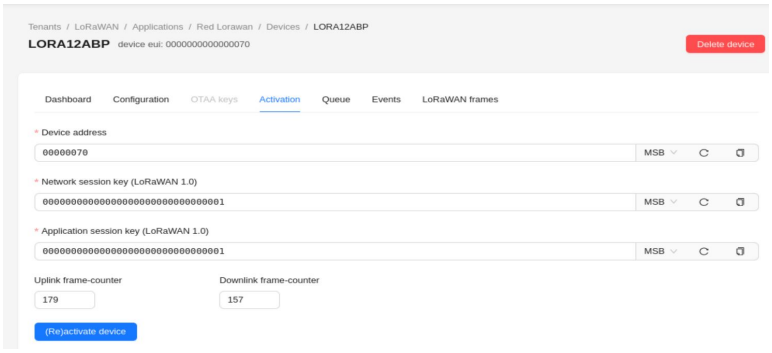


Figura A.4: Configuración de dispositivo en aplicación de ChirpStack (activación)

Cuando ya se tiene configurada la red LoRaWAN en ChirpStack, se enciende la máquina virtual en la que se implementa el *core* 5G y se arranca free5gc:

```
cd free5gc
sudo ./jna_start_all_n3iwf.sh
```

Y, también, se ejecuta el fichero para configurar el enrutamiento de los paquetes de datos LoRaWAN:

```
cd ficheros
sudo ./tablasrutas.sh
```

Ahora, se enciende la máquina virtual donde se implementa el cliente NWu y se arranca:

```
cd NWu-Non3GPP-5GC
sudo ./jna_connect_n3iwf_free5gc.sh
```

Se comprueba que la conexión entre el cliente NWu y el N3IWF se ha hecho correctamente, como se explica en 6.3. Y, se configura el enrutamiento de los paquetes de datos LoRaWAN:

```
cd ficheros
sudo ./nameespaceNWu.sh
```

Una vez hecho esto, ya se puede ver que se han detectado los dispositivos en la organización creada en ChirpStack y los mensajes que llegan al *gateway* encriptados; y, la aplicación desencriptados:

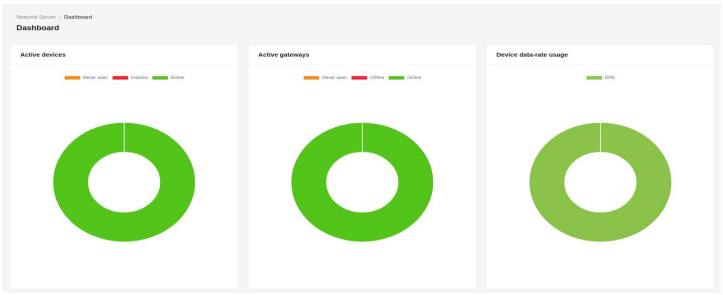


Figura A.5: Detección de dispositivos en ChirpStack

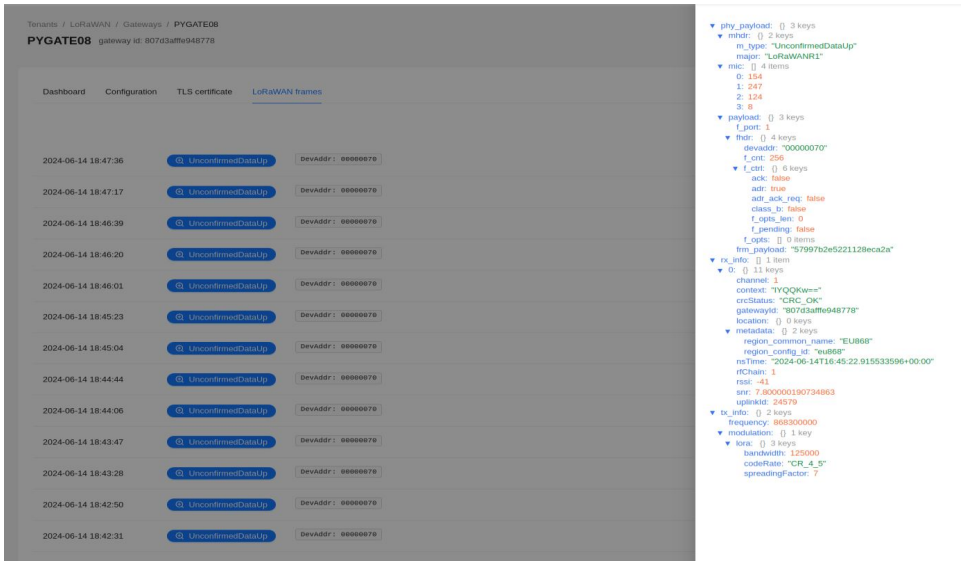


Figura A.6: Mensaje recibido por el *gateway* desde ChirpStack

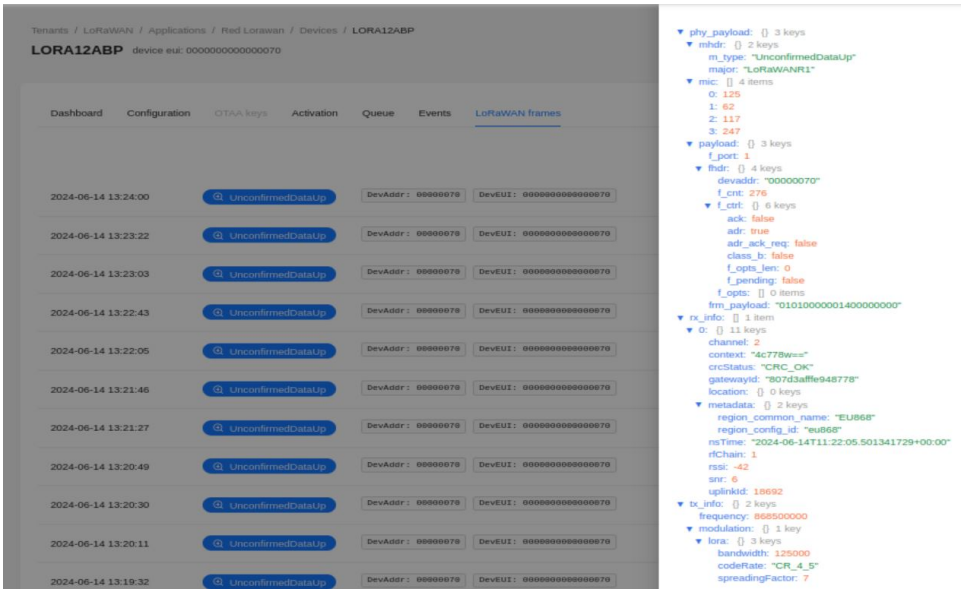


Figura A.7: Mensaje recibido por el servidor LoRaWAN desde la aplicación de ChirpStack

En las distintas interfaces de las máquinas virtuales se pueden obtener datos de los paquetes de datos haciendo uso de *tcpdump* o usando *Wireshark* para futuras pruebas.

Finalmente, cuando se han hecho las pruebas y comprobaciones necesarias, se cierran los servidores y se apagan las máquinas virtuales. Para hacer esto correctamente, en primer lugar se para la conexión del cliente NWu con el N3IWF haciendo uso de *Control+C*, en la terminal en la que se ven sus estados de conexión, y se apaga la máquina. Después, se para el servidor del *core* 5G haciendo uso de *Control+C* en la terminal en la que se ve el estado de sus distintos elementos. Acto seguido, se introduce el comando:

```
sudo ./force_kill.sh
```

Ahora, se apaga la máquina virtual donde se implementa el *core* 5G. Por último, se para el servidor de LoRaWAN introduciendo el siguiente comando y se apaga su máquina virtual:

```
sudo docker-compose down
```

Apéndice B

Instalación y Configuración de ChirpStack

Para realizar la instalación de ChirpStack [56], primero hay que instalar Docker-Compose:

```
sudo apt-get install git docker docker-compose
```

Después, hay que copiar el repositorio en el que se encuentra introduciendo el siguiente comando:

```
git clone https://github.com/chirpstack/chirpstack-docker.git
```

Una vez copiado este repositorio, hay que configurar el MQTT. Para ello, primero tenemos que acceder a la carpeta que se crea al instalar el repositorio anterior (`cd chirpstack-docker`), abrir el archivo `docker-compose.yml` (`sudo nano docker-compose.yml`) e introducir el siguiente comando en la sección de volumes de mosquitto:

```
./configuration/mosquitto/config/mqtt_passwd:/mosquitto/config
    /mqtt_passwd
```

Acto seguido, se genera el fichero con las claves MQTT. Para ello, hay que instalar el *broker mosquitto* usando `sudo apt-get install mosquitto`. Hay que parar y deshabilitar el servicio usando `sudo systemctl disable mosquitto` y `sudo service mosquitto stop`. Después, se genera el fichero en la configuración de *mosquitto*:

```
mosquitto_passwd -c configuration/mosquitto/config/mqtt_passwd
```

Y se introducen el usuario y la contraseña, en este caso el usuario es *MQTTUSER* y la contraseña es *MQTTPASSWORD*.

Ahora, en el fichero `configuration/mosquitto/config/mosquitto.conf`, hay que deshabilitar el usuario anónimo y añadir el path donde se encuentran las claves de MQTT:

```
allow_anonymous false
password_file /mosquitto/config/mqtt_passwd
```

Después, se configura el *gateway-bridge*, en el archivo `chirpstack-gateway-bridge.toml` en `/configuration/chirpstack-gateway-bridge` de la siguiente forma:

```
[integration.mqtt.auth.generic]
servers="tcp://mosquitto:1883"
username="MQTTUSER"
password="MQTTPASSWORD"
```

También, se configura el servidor de red con el archivo `chirpstack-network-server.toml` que se encuentra en `/configuration/chirpstack-network-bridge`:

```
[network_server.gateway.backend.mqtt]
server="tcp://mosquitto:1883"
username="MQTTUSER"
password="MQTTPASSWORD"
```

Y, el servidor de aplicación con el archivo `chirpstack-application-server.toml` que se encuentra en `/configuration/chirpstack-application-bridge`:

```
[application_server.integration.mqtt]
server="tcp://mosquitto:1883"
username="MQTTUSER"
password="MQTTPASSWORD"
```

Finalmente, una vez hecho todos estos pasos, se ejecutan los contenedores para arrancar el servidor:

```
sudo docker-compose up -d
```

En la imagen B.1, se muestra como arranca el servidor:

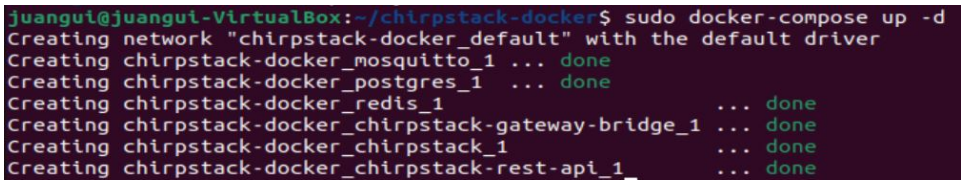


Figura B.1: Creación de los contenedores de ChirpStack

Para parar el servidor, hay que ejecutar:

```
sudo docker-compose down
```

Accediendo a la dirección `http://localhost:8080` se accede a la interfaz de usuario de ChirpStack, desde donde se puede crear una red LoRaWAN y configurarla.

Apéndice C

Instalación de free5gc

Para la instalación de free5gc [57], primero hay que instalar una serie de prerrequisitos. En primer lugar, se comprueba la versión del kernel de la máquina virtual introduciendo el siguiente comando, la versión debe ser la 5.4.0-65-generic, ya que esta herramienta usa, para la implementación del plano de usuario, un GTP 5G que solo está disponible en esa versión:

```
uname -r
```

Después de esto, se instala el lenguaje de programación "Go 1.21.8", en caso de no tenerlo instalado previamente, esto se puede comprobar introduciendo el comando `go version`. Si está instalada otra versión, hay que borrarla usando:

```
sudo rm -rf /usr/local/go
wget https://dl.google.com/go/go1.21.8.linux-amd64.tar.gz
sudo tar -C /usr/local -zxvf go1.21.8.linux-amd64.tar.gz
```

En caso contario, de la siguiente forma se realiza la instalación:

```
wget https://dl.google.com/go/go1.21.8.linux-amd64.tar.gz
sudo tar -C /usr/local -zxvf go1.21.8.linux-amd64.tar.gz
mkdir -p ~/go/{bin,pkg,src}
# The following assume that your shell is bash:
echo 'export GOPATH=$HOME/go' >> ~/.bashrc
echo 'export GOROOT=/usr/local/go' >> ~/.bashrc
echo 'export PATH=$PATH:$GOPATH/bin:$GOROOT/bin' >> ~/.bashrc
echo 'export GO111MODULE=auto' >> ~/.bashrc
source ~/.bashrc
```

Lo siguiente que hay que instalar es "MongoDB"[58]. En este caso, se instala la versión 4.4 para no tener problemas con el *Advanced Vector Extensions* del ordenador personal. Para instalarlo, hay que introducir los siguientes comandos:

```
curl -fsSL https://www.mongodb.org/static/pgp/server-4.4.asc
    | sudo apt-key add -
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt
    /ubuntu/focal/mongodb-org/4.4 multiverse" | sudo tee /etc
    /apt/sources.list.d/mongodb-org-4.4.list
sudo apt update
sudo apt install mongodb-org
```

Y, se arranca:

```
sudo systemctl start mongodb
```

Se instalan los paquetes para la implementación del plano de usuario:

```
sudo apt -y update
sudo apt -y install git gcc g++ cmake autoconf libtool
    pkg-config libmnl-dev libyaml-dev
```

El último prerrequisito es configurar el *host* de red de la máquina virtual:

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o <dn_interface> -j
    MASQUERADE
sudo iptables -A FORWARD -p tcp -m tcp --tcp-flags SYN,RST SYN
    -j TCPMSS --set-mss 1400
sudo systemctl stop ufw
sudo systemctl disable ufw # prevents the firewall to wake up
    after a OS reboot
```

Una vez instalados los prerrequisitos, se instala la carpeta free5gc del siguiente repositorio [59]. En esta carpeta, se encuentra un archivo para ser ejecutado en el *shell* de la máquina virtual que arranca el *core* 5G para poder conectarse con el cliente NWu. También se encuentran los elementos de free5gc en el la carpeta *config_n3iwf*, que hay que configurar correctamente para la conexión con el cliente NWu, con el servidor LoRaWAN y para la interoperabilidad correcta entre sus elementos. Una vez instalada la carpeta, se accede a ella y se implementan todas las funciones del plano de control de free5gc:

```
cd ~/free5gc
make
```

Lo siguiente, es implementar las funciones del plano de usuario. Para ello, se instala el GTP 5G:

```
git clone -b v0.8.6 https://github.com/free5gc/gtp5g.git
cd gtp5g
make
sudo make install
```

Por último, se instala la interfaz web de free5gc. Para hacer esto, hay que instalar *Node.js*:

```
sudo apt remove nodejs -y
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt update
sudo apt install nodejs -y
node -v # check that version is 20.x
sudo corepack enable
```

Se monta el servidor dentro de */free5gc/webconsole*:

```
cd frontend
yarn install
yarn build
rm -rf ../public
cp -R build ../public
```

Para lanzar el servidor de la interfaz web, se usa el siguiente comando dentre de */free5gc/webconsole*:

```
go run server.go
```

Para acceder a la interfaz web, se hace desde la dirección `http://localhost:5000`, y se introduce el usuario *admin* y la contraseña *free5gc*.

Una vez hecho todo esto, se pueden configurar los elementos que componen el *core* 5G accediendo a los archivos que implementan cada uno de ellos en la carpeta */free5gc/config_n3iwf*.

Apéndice D

Instalación de NWu-Non3GPP-5GC client

Para la instalación de NWu-Non3GPP-5GC client [22], hay que instalar previamente una serie de módulos de python3. Primero, se instalan los módulos *card* y *CryptoMobile* de los repositorios [60] y [61]:

```
sudo apt install libpcsclite-dev

pip3 install serial wheel pycrypto  pyscard swig requests
    cryptography

git clone https://github.com/mitshell/card.git
cd card
python3 setup.py install
cd ..

git clone https://github.com/mitshell/CryptoMobile.git
cd CryptoMobile/
python3 setup.py install
cd ..
```

Después, se instala del repositorio [59] la carpeta *NWu-Non3GPP-5GC client*. En esta carpeta, se encuentra un archivo para ser ejecutado en el *shell* de la máquina virtual que arranca el cliente NWu. Este fichero, se llama *jna_connect_n3iwf_free5gc.sh*. En él, hay que configurar correctamente las direcciones IP y los datos de USIM para que realice bien la conexión con free5gc, el proceso de autenticación y la creación del túnel IKEv2/IPSec hasta el N3IWF de free5gc.