TRABAJO FIN DE GRADO

INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Uso de dispositivo LoRaWAN como interfaz IP para un CPE

Autor

Víctor Román García

Directores

Jorge Navarro Ortiz

Natalia Chinchilla Romero

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 16 de Junio de 2025

Uso de dispositivo LoRaWAN como interfaz IP para un CPE

Autor
Víctor Román García

Directores
Jorge Navarro Ortiz
Natalia Chinchilla Romero

Departamento de Teoría de la Señal, Telemática y Comunicaciones

—

Granada, 26 de Junio de 2025

Uso de dispositivo LoRaWAN como interfaz IP para un CPE

Víctor Román García

Palabras clave: LoRaWAN, IP, CPE, MQTT

Resumen

LoRaWAN es una tecnología de red inalámbrica para el IoT (Internet of Things) de tipo LPWAN (Low Power Wide Area Network) que se caracteriza por su largo alcance de cobertura, bajo coste y consumo reducido de energía. Esta tecnología está destinada principalmente para la transmisión de pequeños paquetes de datos procedentes de una gran cantidad de sensores. Los dispositivos finales conectados a los sensores envían los datos mediante la tecnología radio LoRa (Long Range) hacia la pasarela de red LoRaWAN. Ésta, envía los datos encapsulados mediante una red de backhaul TCP/IP hasta los correspondientes servidores de red y aplicación LoRaWAN. Tal como está diseñada la red LoRaWAN, los dispositivos finales no utilizan el protocolo IP por lo que tampoco tienen asignada una dirección IP. En este TFG, se realizará la integración de un dispositivo LoRaWAN como interfaz IP de un CPE (customer premise equipment, que podría ser un móvil inteligente o un PC). Para ello, habrá que configurar el dispositivo LoRaWAN para crear un punto de acceso Wifi al que el CPE sea capaz de enviar tráfico y sea reenviado por LoRaWAN. Y, por otro lado, habrá que implementar un servicio que interactúe con el servidor de aplicación a través de su API para que reenvíe los paquetes recibidos hacia Internet. De esta manera, el CPE podrá enviar y recibir paquetes IP a través del dispositivo LoRaWAN, objetivo último del proyecto.

Usage of a LoRaWAN device as an IP interface for a CPE

Víctor Román García

Keywords:LoRaWAN, IP, CPE, MQTT

Abstract

LoRaWAN is a wireless network technology for the Internet of Things (IoT) based on LPWAN (Low Power Wide Area Network), characterized by its long coverage range, low cost, and reduced energy consumption. This technology is mainly intended for the transmission of small data packets from a large number of sensors. The end devices connected to the sensors send data using LoRa (Long Range) radio technology to the LoRaWAN network gateway. The gateway then forwards the encapsulated data via a TCP/IP backhaul network to the corresponding LoRaWAN network and application servers. As designed, the LoRaWAN network does not use the IP protocol for end devices, so they are not assigned IP addresses.

In this Bachelor Thesis, the integration of a LoRaWAN device as an IP interface for a CPE (Customer Premises Equipment, such as a smartphone or PC) is the main goal. To achieve this, the LoRaWAN device must be configured to create a Wi-Fi access point that the CPE can use to send traffic, which will then be forwarded via LoRaWAN. Additionally, a service will need to be implemented to interact with the application server through its API in order to forward the received packets to the Internet. In this way, the CPE will be able to send and receive IP packets through the LoRaWAN device, which is the ultimate goal of the project.

Yo, **Víctor Román García**, alumno de la titulación Ingeniería de Tecnologías de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76740370L, 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: Víctor Román García

Granada a 16 de Junio de 2025.

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

D. **Natalia Chinchilla Romero**, Estudiante de Doctorado en el Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Uso de dispositivo LoRaWAN como interfaz IP para un CPE***, ha sido realizado bajo su supervisión por **Víctor Román García**, 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 16 de Junio de 2025 .

Los directores:

Jorge Navarro Ortiz **Natalia Chinchilla Romero**

Agradecimientos

A mis padres, por toda la confianza y el cariño; a Mario, por ser el mejor consejero que un hermano pequeño puede tener; a mis amigos Jorge, Antonio, David, Anabel, Pedro, Martín, Alejandro, José y Andrés, por acompañarme en todo momento; y a Jorge y Natalia, por la paciencia y la ayuda prestada.

Índice general

1. Introducción **1**
1.1. Contexto y motivación 1
1.2. Objetivos 2
1.3. Estructura 3

2. Estado del Arte **5**

3. Planificación y Coste **9**
3.1. Planificación Temporal 9
3.2. Planificación de Recursos 12
3.2.1. Recursos Hardware 12
3.2.2. Recursos Software 13
3.2.3. Recursos Humanos 14
3.2.4. Coste Total 15

4. Fundamentos Teóricos **17**
4.1. LoRa 17
4.1.1. Fundamentos de LoRa 17
4.1.2. Banda ISM 19
4.2. LoRaWAN 19
4.2.1. Arquitectura de red 20
4.2.2. Clases de LoRaWAN 21
4.2.3. Autenticación y métodos de conexión 22
4.2.4. Uso de ADR 23
4.2.5. Protocolo MQTT 24
4.2.6. Arquitectura Chirpstack 25
4.2.7. Limitaciones generales 26

5. Diseño e Implementación **29**
5.1. Diseño del flujo de comunicación 29
5.1.1. Uplink 29
5.1.2. Downlink 30
5.2. Servidor 30
5.3. Forwarder 32

5.3.1. Fragmentación de Paquetes IP 32
5.3.2. SNAT 33
5.3.3. Funciones asociadas a MQTT 34
5.3.4. Funciones asociadas a gRPC API 34
5.3.5. Sniffer Ethernet 35
5.4. Dispositivo LoRaWAN 35
5.4.1. Creación del AP 35
5.4.2. Funciones Wifi 36
5.4.3. Configuración LoRaWAN 37
5.5. Resto de configuración 38

6. Evaluación de la Implementación 39
6.1. ICMP 40
6.1.1. Sin fragmentación 40
6.1.2. Con fragmentación 45
6.2. UDP y TCP 50

7. Conclusiones y Trabajos Futuros 57
7.1. Conclusiones 57
7.2. Trabajos futuros 58

A. Anexo I. Manual de Instalación 59

B. Anexo II. Manual de Usuario 63

Bibliografía 64

Índice de figuras

1.1. Estimación del mercado global IoT hasta 2030 [1] 2

2.1. Escenario de Ip2LoRa de Aribus Cibersecurity 6
2.2. Arquitectura Ip2LoRa de Airbus Cibersecurity 6
2.3. Arquitectura de red LoRaWAN para comunicación con protocolo IPv6 8

3.1. Diagrama de Gantt 11
3.2. Mota Heltec Wireless Stick V3 13
3.3. IMST Lite Gateway 13

4.1. Espectrograma de una transmisión LoRa [18] 18
4.2. Distribución de canales en la región EU 863-870MHz (8 canales) para LoRaWAN [21] 19
4.3. Arquitectura común de red LoRaWAN [22] 20
4.4. Esquema de comunicación para autenticación en OTAA [23] . 23
4.5. Arquitectura típica de Chirpstack con diferentes conexiones a las puertas de enlace 25

5.1. Flujo de comunicación en el uplink y compontes involucrados 30
5.2. Flujo de comunicación en el downlink y componentes involucrados 30
5.3. Configuración básica para la creación de un perfil 31

6.1. *Log* del *forwarder* para *ping* de tamaño 90 bytes 41
6.2. Serial monitor de Arduino IDE para dispositivo LoRaWAN al envío y recepción del *echo-request* y *echo-reply* 42
6.3. Paquete IP en el *downlink* decodificado 43
6.4. Histograma (azul) de los *pings* realizados para tamaño 90 bytes y densidad de probabilidad estimada (naranja) 43
6.5. Histograma (azul) de los *pings* realizados para tamaño 200 bytes y densidad de probabilidad estimada (naranja) 44
6.6. *Log* del *forwarder* aplicando fragmentación con tamaños de los fragmentos resaltados en color verde y gris 46

6.7. Serial monitor del dispositivo LoRaWAN aplicando fragmentación 46
6.8. Histograma (azul) de los *pings* realizados para tamaño 500 bytes y densidad de probabilidad estimada (naranja) 47
6.10. Histograma (azul) de los *pings* realizados para tamaño 1500 bytes y densidad de probabilidad estimada (naranja) 48
6.9. Serial monitor del dispositivo LoRaWAN aplicando y recibiendo fragmentación para 1500 bytes 49
6.11. Configuración cliente UDP desde la aplicación *TCP UDP Server and Client* 51
6.12. Lanzamiento del servidor y cliente en la máquina anfitriona (arriba servidor, abajo cliente). 51
6.13. Intercambio de mensajes por chat UDP desde el móvil 52
6.14. *Log* del *forwarder* en el intercambio de mensajes a través de un chat basado en UDP 53
6.15. Envío de tramas LoRaWAN 54
6.16. Lanzamiento del servidor y cliente en la máquina anfitriona (arriba servidor, abajo cliente). Establecimiento conexión TCP. 55
6.17. *Log* del *forwarder* en el envío de mensaje a través de un chat basado en TCP. En el cuadro rojo se marca el envío downlink para indicarle a *lorawan_user* que su mensaje ha llegado . . . 55

A.1. Gateway configurado y disponible en plataforma Chirpstack . 60
A.2. Trama LoraWAN (Join Request) mandada por el dispositivo LoRaWAN a la plataforma Chirpstack 61

B.1. Serial monitor del dispositivo LoRaWAN. Envía un trama LoRaWAN para Join Request y crea el punto de acceso wifi, el dispositivo móvil se une y le asigna la dirección 192.168.4.2 64
B.2. Uso de cortafuegos mediante NetGuard. Se permite el envío de tráfico únicamente para las aplicaciones que se utilizarán en las pruebas. 65
B.3. *Ping* realizado con un dispositivo móvil a la dirección IP del *gateway* como prueba de funcionamiento 66

Índice de cuadros

3.1. Planificación y duración de las tareas 12
3.2. Tiempo invertido aproximado en el proyecto 15
3.3. Precio por unidad de los recursos hardware 15
3.4. Precio por licencia de los recursos software 15
3.5. Precio por hora de los recursos humanos 15
3.6. Suma total del coste asociado a los recursos 16

6.1. Direcciones IP relevantes 40
6.2. Resumen de resultados de las pruebas ICMP 50

Capítulo 1

Introducción

1.1. Contexto y motivación

Las comunicaciones inalámbricas llevan en auge desde las últimas décadas y esta evolución no parece frenarse. La importancia de este tipo de tecnología es cada vez mayor, especialmente en entornos industriales e incluso domésticos. La razón es lógica, una evolución tecnológica busca alternativas donde se puedan hacer tareas de forma más rápida, de manera más cómoda y/o con menor coste.

IoT (Internet over Things) constituye esta red de dispositivos conectados de forma inalámbrica con el objetivo de proporcionar y recoger información entre dispositivos en un entorno determinado. Datos ofrecidos por IoT Analytics en la figura 1.1 muestra de forma estimada que hasta 18 mil millones de dispositivos en 2024 en el mundo utilizaban conexiones IoT, lo que supone el doble de conexiones que existían en 2019 y cerca de cuatro veces más las que se esperan en 2030 comparadas al primer año de la gráfica.

Dentro de este incremento en las conexiones IoT destaca LPWAN (Low Power Wide Area Connector) como el segundo mayor porcentaje de uso de conexiones IoT después de la popular tecnología 5G. Este tipo de conexiones encajan a la perfección en entornos IoT, donde el número de conexiones es muy alto pero el tamaño de la información compartido es mínimo. Por ejemplo, el tipo de información típico que se comparte en entornos IoT son estadísticas del entorno que han captado ciertos sensores, o también mensajes de estado del dispositivo.

LoRaWAN, que es la tecnología utilizada en este proyecto en concreto, es una especificación dentro de las redes LPWAN como lo es también SigFox. Ambas especificaciones son capaces de ofrecer redes de gran cobertura y poco consumo y cumplen con las características que las redes de entornos

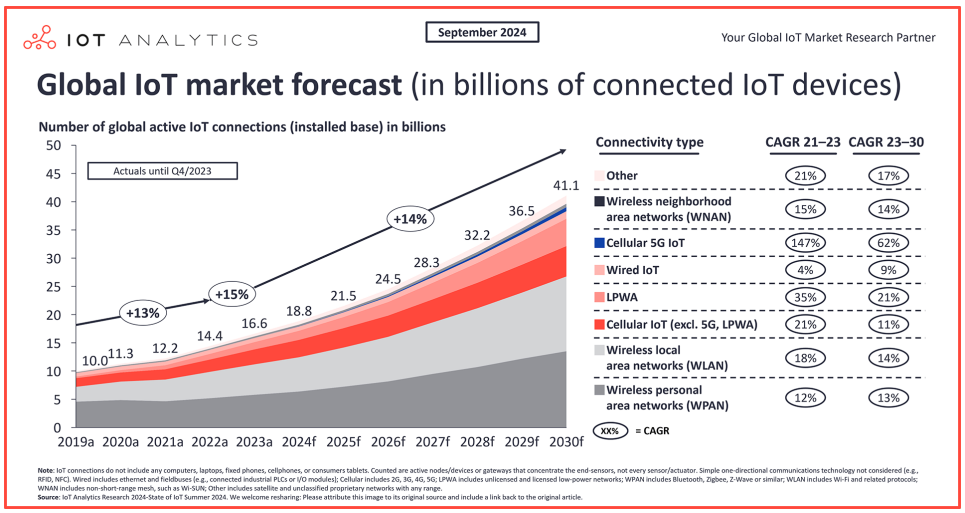

Figura 1.1: Estimación del mercado global IoT hasta 2030 [1]

IoT buscan. Las diferencias entre una y otra especificación son sutiles, pero las ventajas de LoRaWAN frente a otras especificaciones son las de permitir la bidireccionalidad de los dispositivos, capacidad de desplegar redes privadas, ofrecer un mayor ancho de banda que SigFox y mayor escalabilidad de dispositivos.

Como se ha repasado, el número de este tipo de conexiones sigue en aumento, pero el uso de estas conexiones está limitado a los dispositivos que únicamente son capaces de utilizar dicha tecnología. En este proyecto se pretende ofrecer una solución para conectar un CPE común a dispositivos LoRaWAN, de forma que sea capaz de enviar información a través de estas redes, como pueden ser mensajes de auxilio del usuario en caso de emergencia en zonas sin cobertura móvil (por ejemplo, entornos rurales y/o montañosos). La mayoría de CPEs son capaces de detectar puntos de acceso de red IP, así que la elección de este tipo de solución es natural.

1.2. Objetivos

El objetivo de este proyecto es el de poder crear un flujo de comunicación entre CPEs integrados en una arquitectura de red LoRaWAN, manipulando el tráfico IP de los dispositivos para poder ser transportado por LoRaWAN. En concreto, los objetivos del proyecto son:

- Comprender en profundidad dentro de las redes LPWAN la especificación LoRaWAN, su configuración, funcionamiento, capacidades y limitaciones.

- Montar una red LoRaWAN con un servidor de red por donde se puedan comunicar este tipo de dispositivos.
- Crear un método de conexión a dispositivos y redes LoRaWAN para CPEs como puede ser un dispositivo móvil u ordenador.
- Establecer un método de envío de ida y vuelta de información desde los CPEs hacia esta red.

1.3. Estructura

A continuación se presentan los capítulos del documento y qué se tratará en cada uno de ellos.

- **Capítulo 1. Introducción:** Se da a conocer un primer contexto de los objetivos del proyecto, la tecnología utilizada y el entorno donde este proyecto es trascendente.
- **Capítulo 2. Estado del arte:** Se comenta el estado actual de proyectos similares, estado del proyecto, soluciones alternativas si existen y en general cómo de desarrollada se encuentran las tecnologías implementadas.
- **Capítulo 3. Planificación y coste**: Se describe una aproximación de la duración y el coste total del proyecto, presentando las tareas necesarias para su realización y todos los recursos disponibles.
- **Capítulo 4. Fundamentos teóricos**: Se revisan los conocimientos teóricos que involucran el proyecto, como la tecnología utilizada, protocolos de comunicación y arquitecturas tipo relacionadas al proyecto y sus componentes.
- **Capítulo 5. Diseño e Implementación:** Se presenta la idea de diseño para poder llevar a cabo la finalidad del proyecto, así cómo una descripción de su implementación final.
- **Capítulo 6. Evaluación de la Implementación:** Se analizan los resultados obtenidos de la implementación propuesta en diferentes escenarios.
- **Capítulo 8. Conclusión y Trabajos Futuros:** Se repasan los resultados obtenidos tras la realización del proyecto y se hace un balance final de los mismo. Gracias a estas conclusiones, se valoran posibles implementaciones y trabajos futuros para poder mejorar el proyecto.

Capítulo 2

Estado del Arte

En este apartado se presentan y analizan trabajos con una finalidad parecida a los de este proyecto con el fin de entender mejor qué se ha conseguido, cómo se ha realizado y qué mejoras o alternativas se pueden realizar.

En la actualidad, la opción mayormente citada es la de la empresa de ciberseguridad para uso militar Airbus CiberSecurity, que se denomina IP2LoRa [2]. Disponen de un repositorio en GitHub en el cual se ofrece código Python para una configuración determinada de una red LoRa y poder enviar tráfico IP a través de la misma. Ofrece además una opción de posibilidad de compresión de cabeceras IP, utilizando el protocolo ROHC. Nótese que esta es una solución para redes LoRa y realmente no guarda una relación directa con el entorno de este proyecto que es para LoRaWAN, pero es igualmente uno de los proyectos más citados en este ámbito.

A pesar de ser una solución fácil y de libre uso, la propia empresa advierte de que se trata de una prueba de concepto y que en la actualidad su solución sufre de diversas limitaciones, especialmente la gran pérdida de paquetes debido a la pobre coordinación entre LoRa y TCP.

Este sistema está probado en tres dispositivos LoRa: el B-L072Z-LRWAN1, WisDuino [3] y LoStick [4], siendo todos ellos dispositivos más caros que el utilizado en este proyecto.

El escenario que se contempla en el proyecto de Airbus Cibersecurity incluye comunicación directa LoRa pero de máquinas que se encuentran en la misma subred, como se aprecia en la figura 2.1. Es decir, propone comunicación de máquinas en protocolo TCP/IP mediante dispositivos LoRa que se comunican entre ellos directamente sin el uso de un servidor de red.

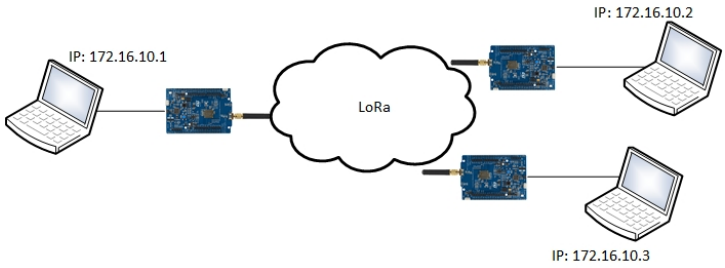

Figura 2.1: Escenario de Ip2LoRa de Aribus Cibersecurity

Por otra parte, tiene como ventaja la mayoría de implementaciones se encuentran en las máquinas que se quieren comunicar y no en los dispositivos LoRa, despejando las funcionalidades y la carga de los mismos. Estas implementaciones en máquina las denominan IP2LoRa Stack (2.2), e incluyen la creación de una interfaz vacía para asignar una dirección IP válida a la comunicación LoRa, así como una dirección MAC falsa. El tráfico de esta interfaz es captado y manipulado mediante la librería Scapy, que a través de la conexión directa con los dispositivos LoRa, manda los paquetes IP ya comprimidos y cifrados con un simple XOR (método de cifrado comúnmente utilizado por su simplicidad).

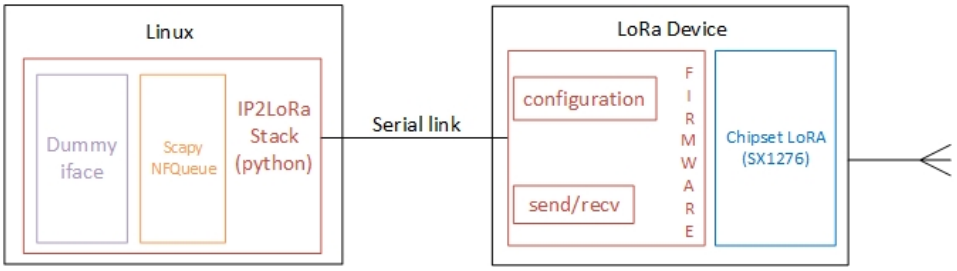

Figura 2.2: Arquitectura Ip2LoRa de Airbus Cibersecurity

En resumen, las desventajas de este sistema principalmente se deben al escenario propuesto. Mientras que sirve como un buen concepto para la comunicación IP a través de LoRa, se trata de un proyecto con un escenario muy descentralizado donde cada máquina debe tener instalado las herramientas necesarias para poder correr el escenario, y sobre todo, que no hace uso de una arquitectura LoRaWAN propia. Una implementación similar pero mediante un servidor de red que sea éste el que ejecute este tipo de acciones sería más apropiado para un uso más general y controlado, esto se puede realizar utilizando una arquitectura LoRaWAN.

Otro artículo busca mejorar el rendimiento de este proyecto anterior [5]. Airbus Cibersecurity destacaba en su proyecto el número de pérdidas

y colisiones que se creaban debido a que TCP no era consciente de las limitaciones de LoRa.

Por ello, en este artículo proponen un cambio sutil en la configuración de las cabeceras TCP en el IP2LoRa Stack. Este cambio se centra en el campo de TCP *awnd* o advertised window, que indica cuántos bytes es capaz de recibir el receptor sin ningún desbordamiento. Si este campo valiese 1, la comunicación se vería obligada a realizarse esperando a que llegase un sólo paquete y se confirmase antes de seguir.

Es una mejora interesante porque a pesar de tener como consecuencia hacer el sistema de comunicación mucho más lento, esto era precisamente una de las características ya propias de las comunicaciones LoRa, y a cambio, soluciona uno de los problemas que existía en el proyecto propuesto por Airbus Cibersecurity de IP2LoRa. Igualmente, tampoco plantea una implementación en una arquitectura LoRaWAN, algo que sí se realiza en este proyecto.

Finalmente, la propia empresa SemTech ha propuesto una implementación de comunicación IPv6 a través de LoRaWAN [6], donde el principal cambio se encuentra en la inclusión de compresión y fragmentación por SCHC (función similar a ROHC anteriormente).

SCHC es un mecanismo de compresión y fragmentación que funciona muy bien en IPv6 gracias a que este último dispone de campos fijos y predecibles, tal y como lo describe SemTech. Sabiendo de antemano qué se va a transmitir gracias a la naturaleza de las redes IoT hace que estos mecanismos de compresión sean aún más eficaces. En la figura 2.3 se encuentra un ejemplo de esta implementación de red, donde los dos activos más novedosos son la inclusión de SCHC para la compresión y fragmentación inicial en el dispositivo final LoRaWAN, y el de una puerta de enlace SCHC que reconstruye los paquetes IPv6 para posteriormente procesarlos de forma apropiada.

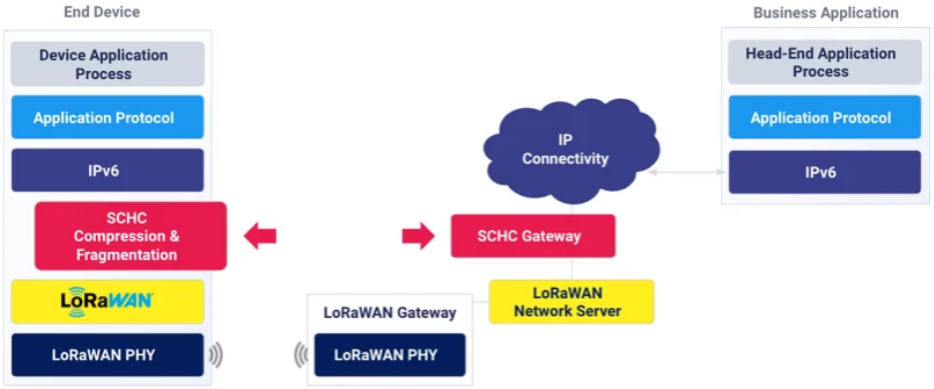

Figura 2.3: Arquitectura de red LoRaWAN para comunicación con protocolo IPv6

Esta implementación es novedosa y bastante interesante, posibilitando comunicación IP/IPv6 a través de LoRaWAN y siguiendo un diseño e implementación oficial de la propia SemTech. Lo que no ha incluido SemTech es una implementación para IP/IPv4, que a pesar de no ser prometedora para el futuro, se sigue utilizando aún ampliamente. Además los protocolos que se soportan en esta implementación son CoAP (*Constrained Application Protocol*) o similares, que no son protocolos de uso común. Es por esto que el diseño de este proyecto está pensado para IPv4, que hasta ahora no se ha desarrollado de forma oficial.

Capítulo 3

Planificación y Coste

En este capítulo se describirá cómo se ha llevado a cabo el proyecto a lo largo del tiempo, las diferentes tareas necesarias para completarlo y todo los tipos de recursos empleados.

3.1. Planificación Temporal

Se enumerarán las diferentes tareas y fases del proyecto, ordenadas de forma cronológica para poder reconstruir los pasos realizados en la conformación del trabajo.

1. **Estudio y comprensión de objetivos**
    - Revisión de las tecnologías LoRa/LoRaWAN: Implica el estudio y repaso la tecnología de comunicación LoRa y su arquitectura de red en LoRaWAN, sus características y en concreto, familiarizarse con los términos clave.
    - Familiarizarse con los recursos disponibles: Comprender de qué son capaces los dispositivos que se van a utilizar en el proyecto de manera temprana puede facilitar fases de diseño e implementación posteriores.
    - Comprensión de los objetivos del proyecto: Es importante entender más allá de una prueba de concepto los objetivos finales del proyecto y entender realmente qué se quiere conseguir y se puede entender como un resultado positivo.
    - Repaso del estado del arte: Una vez familizarizado con los objetivos del proyecto, se realiza una revisión de proyectos con objetivos similares o arquitecturas parecidas para poder entender cuál es el punto de partida.

2. **Diseño e implementación**

   - Instalación de las herramientas necesarias: Como preparación para el montaje de una arquitectura de red LoRaWAN. Se instalan las herramientas necesarias en la máquina donde se vaya a realizar el proyecto.
   - Montaje de una red LoRaWAN estándar: Una vez instaladas las herramientas necesarias y se tienen a disposición los dispositivos necesarios, se monta la red y se comprueba el funcionamiento básico de la misma.
   - Diseño de comunicación uplink: Se prepara e idea la comunicación desde un CPE hasta el destinatario. Esto involucra diseñar la comunicación de CPE con dispositivo LoRaWAN o cómo el CPE mandará información a este dispositivo inicialmente, además de entender cómo el servidor manejará la información recibida y poder reenviarla al destino.
   - Diseño de comunicación downlink: Una vez ideado cómo llevar a cabo la comunicación uplink del proyecto, se debe diseñar el camino de vuelta, deshaciendo los pasos anteriores e implementando nuevos si es necesario.
   - Implementación de los componentes y las configuraciones necesarias: Se preparan los componentes necesarios para hacer que la arquitectura de red básica anterior realice los objetivos del proyecto. Esto implica la instalación de estos componentes, ajustarlos y configurarlos apropiadamente.

3. **Evaluación de resultados en diferentes escenarios**

   - Comprobación de la implementación: Se evalúan los primeros resultados obtenidos tras la fase de diseño e implementación, y si es necesario se vuelven a ajustar configuraciones.
   - Testeo en diferentes escenarios: Se preparan nuevos escenarios de prueba para obtener mayor información de las capacidades del diseño.
   - Evaluación final: Se realiza un balance de aspectos positivos y negativos de los resultados obtenidos y se buscan posibles mejoras futuras.

4. **Escritura de la memoria del proyecto**

   Se lleva a cabo durante la elaboración del trabajo para poder ir documentar la realización y los resultados de cada fase anteriormente descrita.

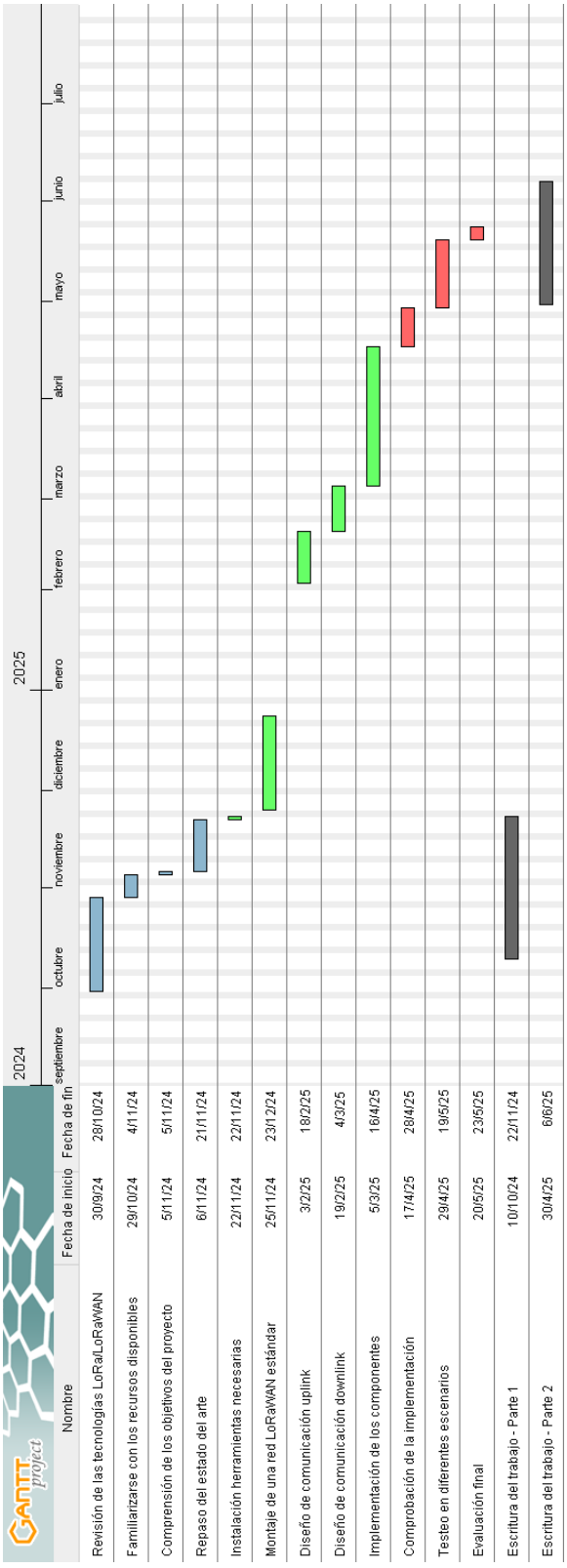

Figura 3.1: Diagrama de Gantt

En la siguiente tabla se presenta a modo de resumen la planificación temporal de las tareas y su duración.

Tareas	**Duración (días)**
Revisión de las tecnologías LoRa/LoRaWAN	20
Familiarizarse con los recursos disponibles	7
Comprensión de los objetivos del proyecto	1
Repaso del estado del arte	15
Instalación herramientas necesarias	1
Montaje de una red LoRaWAN estándar	30
Diseño de comunicación uplink	15
Diseño de comunicación downlink	15
Implementación de los componentes	45
Comprobación de la implementación	10
Testeo en diferentes escenarios	21
Evaluación final	5
Escritura del trabajo	60

Cuadro 3.1: Planificación y duración de las tareas

Esto resulta en un total de 244 días, que suponiendo una media de 3 horas de trabajo cada día, se obtienen unas 750 horas en total.

3.2. Planificación de Recursos

A continuación se describen los recursos utilizados de cada tipo y un cálculo del coste total del proyecto.

3.2.1. Recursos Hardware

Referidos a todos los recursos físicos de hardware como dispositivos o materiales utilizados.

- **Portátil de uso personal**: Se trata de un Lenovo ThinkPad X1 Carbon 12th Gen con un procesador Intel Core Ultra 7, 32 GB de RAM, gráfica Intel Graphics y una memoria SSD de 512 GB.
- **Dispositivo móvil Xiaomi Redmi 6**: Utilizado como CPE de prueba en el proyecto, aunque no es estrictamente necesario porque el mismo ordenador personal podría servir también.
- **Dispositivo LoRaWAN**: Es un Heltec Wireless Stick V3 [7] que contiene un chip ESP32 en su interior y se ha elegido por su com-

patibilidad con Arduino y la integración de Wifi, LoRa y Bluetooth. Se encarga de crear un punto de acceso Wifi y reenviar los paquetes recibidos a la red LoRaWAN.

Figura 3.2: Mota Heltec Wireless Stick V3

- **Gateway o Puerta de enlace**: Se trata de una IMST Lite Gateway [8], y genera la red inalámbrica LoRaWAN mientras se comunica directamente con el servidor para proporcionarle toda la información recogida de la red y las motas que se conectan a dicho gateway.

Figura 3.3: IMST Lite Gateway

- **Adaptador USB a Ethernet**: No es estrictamente necesario pero facilita la conexión directa de la gateway con la máquina anfitriona servidor [9].

3.2.2. Recursos Software

Son las herramientras software utilizadas y necesarias para poner en funcionamiento la arquitectura de red. Todos son software libre por tanto,

su coste de uso es nulo.

- **Arduino IDE 2.3.4**: Software necesario para la programación del chip ESP32 del dispositivo LoRaWAN y su configuración [10].
- **Python**: Utilizado como lenguaje de programación para ciertos componentes en el proyecto [11].
- **Chirpstack Platform**: Plataforma de código abierto necesaria para la creación de la red LoRaWAN, ofreciendo los componentes software necesarios [12].
- **Virtual Box 7.0**: Es un software para la virtualización de sistemas operativos. En él, se ha virtualizado una máquina Ubuntu 22.0 donde el servidor Chirpstack ha sido desplegado [13].
- **Gantt Project**: Utilizado como herramienta para la creación del diagrama temporal para las tareas asociadas al proyecto [14].
- **Putty**: Es un software que sirve como cliente SSH y telnet y es utilizado para poder acceder al gateway desde el portátil de uso personal. Es de código abierto [15].
- **Overleaf**: Es la plataforma online utilizada para la escritura de la memoria del trabajo mediante el lenguaje Latex [16].

3.2.3. Recursos Humanos

Son las personas involucradas en la realización del proyecto, así como el tiempo de inversión aproximado que cada uno de los integrantes ha invertido en el mismo. No deja de ser una aproximación basada en el número de horas invertidas en cada una de las tareas del proyecto por parte del alumno, y del número de tutorías y trabajo externo que los tutores han realizado.

Las personas involucradas son:

- **Víctor Román García (Alumno)**: Estudiante de la Universidad de Granada, en el grado de Ingeniería de Tecnologías de la Telecomunicación.
- **Jorge Navarro Ortiz (Tutor)**: Profesor Titular en la Universidad de Granada del departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.
- **Natalia Chinchilla Romero (Co-Tutora)**: Estudiante de Doctorado en el Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Nombre	**Horas invertidas**
Víctor Román García	400
Jorge Navarro Ortiz	60
Natalia Chinchilla Romero	20

Cuadro 3.2: Tiempo invertido aproximado en el proyecto

3.2.4. Coste Total

Por último, se muestra el coste estimado final del proyecto en las siguientes tablas. Los recursos hardware van acompañados del porcentaje de uso útil en el proyecto, seguido del coste de amortización.

Recursos Hardware	**Precio/Unidad (€)**
Portátil personal (16 %)	250
Dispositivo Móvil (5 %)	10
Dispositivo LoRaWAN (30 %)	5
Gateway (20 %)	40
Adaptador USB a Ethernet (50 %)	5

Cuadro 3.3: Precio por unidad de los recursos hardware

Por lo tanto el coste total en recursos hardware es de 310€.

Recursos Software	**Precio/Licencia(€)**
Arduino IDE	0
Python	0
Chirpstack Platform	0
Virtual Box	0
Gantt Project	0
Putty	0
Overleaf	0

Cuadro 3.4: Precio por licencia de los recursos software

El coste total en recursos software es nulo debido al uso íntegro de recursos de software libre.

Recursos Humanos	**Precio/Hora(€)**
Víctor Román García	25
Jorge Navarro Ortiz	50
Natalia Chinchilla Romero	30

Cuadro 3.5: Precio por hora de los recursos humanos

Para el presupuesto del alumno, se ha estimado el sueldo promedio de un ingeniero júnior, cercano a los 25 €la hora. El coste total por recursos humanos teniendo en cuenta las horas de la tabla 3.2 es de 13.600 €.

Una suma total de todos los recursos y el cálculo final del coste total del proyecto se encuentra en la tabla 3.6, dando una suma de 13.910 €.

Recursos	**Coste/Recurso(€)**
Hardware	310
Software	0
Humano	13.600
Total	13.910

Cuadro 3.6: Suma total del coste asociado a los recursos

Capítulo 4

Fundamentos Teóricos

En esta sección se describen los fundamentos básicos y teóricos necesarios para la realización del proyecto, especialmente los que comprenden la tecnología utilizada.

4.1. LoRa

4.1.1. Fundamentos de LoRa

Lo primero es distinguir LoRa de LoRaWAN. Hasta el momento se ha hablado del uso en el proyecto de LoRaWAN como tecnología de comunicación inalámbrica entre dispositivos IoT, pero realmente LoRaWAN es el protocolo de comunicación que hace uso de LoRa como tecnología en una infraestructura de red. Se puede entender en nuestro proyecto como es LoRa representante de la capa física.

LoRa fue desarrollada por Cycleo con patente 9647718-B2 [17], empresa posteriormente adquirida por SemTech, y se puede definir como una técnica de modulación por espectro ensanchado, en concreto, utiliza una técnica por frecuencia modulada pulsada o *chirping* (CSS). Este tipo de modulaciones eran comunes en aplicaciones para radares antes de usarse en LoRa. CSS hace un barrido de frecuencias con una duración definida (conocido como periodo de chirp) para ensanchar la señal. Gracias a estas características, CSS es capaz de ofrecer mejor resistencia a efectos Doppler que otras técnicas, menor complejidad de recepción de la señal y mejor resistencia frente a efectos multipath, además de distribuir la energía de la señal será distribuida por el ancho de banda.

Un espectrograma común de una transmisión LoRa es el de la figura 4.1. Comúnmente una transmisión LoRa contiene 8 chirps "preamble"para sincronización de sistema, más chirps previamente definidos para indicar el

final del preamble y el inicio de la transmisión de datos (SFD o Start Frame Delimiter) y posteriormente la información en sí. Cada señal chirp puede ser de tipo Up-chirp o Down-chirp, dependiendo de si la frecuencia aumenta o disminuye para cada periodo de señal. Es también común enviar en paquetes LoRa además de una respectiva cabecera, unos bits de CRC como método de detección de errores.

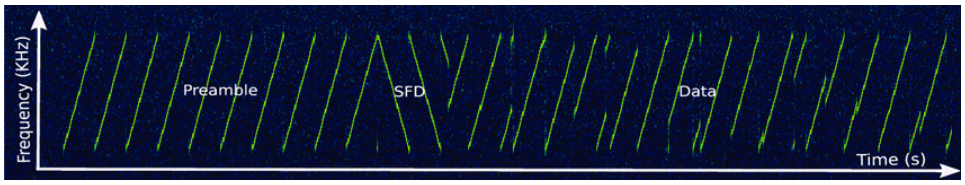

Figura 4.1: Espectrograma de una transmisión LoRa [18]

Este tipo de modulación conlleva el uso de ciertos parámetros de gran importancia, uno de ellos siendo especialmente relevante como es el Spreading Factor o SF. De este valor y el ancho de banda (BW) depende la duración de la señal chirp (4.1).

Por lo tanto, asumiendo un ancho de banda fijo, se puede estimar la tasa de datos para diferentes valores de SF (4.2). En esta ecuación, CR se corresponde al Code Rate o a la tasa de código, que viene dada por el número de bits de datos útiles frente al número de bits transmitidos[1]. Un mayor CR puede mejorar la robustez y reducir los fallos en el envío a costa de una menor tasa de datos, por eso también es un parámetro necesario para el cálculo de la ecuación 4.2.

$$T_s = 2^{SF}/BW \tag{4.1}$$

$$R_b = SF * T_s^{-1} * (\frac{4}{4+CR}) \tag{4.2}$$

De esta ecuación se puede determinar la importancia del SF en comunicación LoRa, ya que no sólo determina la tasa de envío de datos si no también el rango de cobertura y la sensibilidad en la recepción. La modulación LoRa utiliza comúnmente SF de 7 chirps por cada bit de datos (mejor tasa de datos pero peor sensibilidad en recepción y menor rango de cobertura) hasta SF de 12 chirps para cada bit de datos (menor tasa de datos pero gran alcance).

[1]LoRa utiliza como codificación de corrección de errores FEC, añadiendo bits extra en el envío de un paquete para detectar y corregir errores en la recepción.

4.1.2. Banda ISM

Es necesario conocer qué rangos de frecuencia están habilitados legalmente para el uso de LoRa y LoraWAN. Las bandas ISM son bandas de radio para uso tanto industrial, médico o científico pero excluyen las aplicaciones de telecomunicaciones generales. Igualmente, según la ITU (Unión Internacional de Telecomunicaciones) se puede hacer uso de dichas bandas sin licencia cumpliendo ciertos requisitos [19].

Para LoRaWAN en Europa, la banda disponible para su uso es la ISM EU863-870 MHz (popularmente conocida como la EU868 MHZ), con una potencia máxima permitida de hasta +14 dBm. Hay distintas sub-bandas para zonas específicas que no sólo incluyen las regiones europeas. Para Europa existen dos regiones de radiofrecuencia que permiten los 8 canales o los 16 canales [20], en ambas regiones separados 200 kHz. Típicamente se utiliza la región para 8 canales que se distribuyen como en la figura 4.2 y son:

- 8 canales LoRa de 125 kHz de BW, desde 250 bps hasta 5.5 kbps.
- 1 canal LoRa de 250 kHz de BW, hasta 11 kbps.
- 1 canal FSK de 250 kHz de BW, hasta 50 kbps.

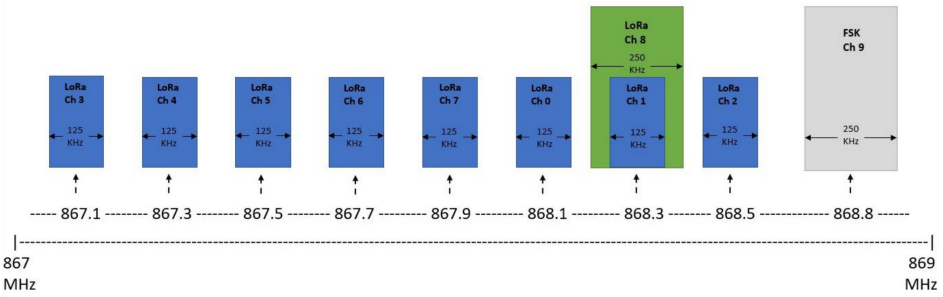

Figura 4.2: Distribución de canales en la región EU 863-870MHz (8 canales) para LoRaWAN [21]

4.2. LoRaWAN

Como se comentaba al principio de la anterior sección, si LoRa representa la capa física de nuestra red, LoRaWAN conforma la capa MAC y define la arquitectura de red. Se trata de una especificación o estándar establecido por LoRaWAN Alliance y es también el estándar utilizado en el proyecto.

Es vital esta arquitectura para dar uso a la técnica de modulación LoRa y garantizar los aspectos necesarios para entornos IoT, como el consumo de recursos, seguridad y en general toda configuración de la red.

4.2.1. Arquitectura de red

En una arquitectura de red LoRaWAN común destacan cuatro principales actores en una jerarquía formada por varias topologías estrella (mejor conocida como Star-of-Stars) tal y como se presenta en la figura 4.3. Estos actores son los dispositivos finales o end nodes (pudiendo ser sensores y/o actuadores), gateways y servidores de red y de aplicación.

Los dispositivos finales aprovechan las características de largo alcance de LoRa y se conectan a uno o varios gateways de forma directa. Éstos a su vez son los que tienen conexión directa con el servidor de red mediante TCP/IP mientras que el servidor de red es el responsable de gestionar el tráfico entre los dispositivos o nodos finales y los servidores de aplicación.

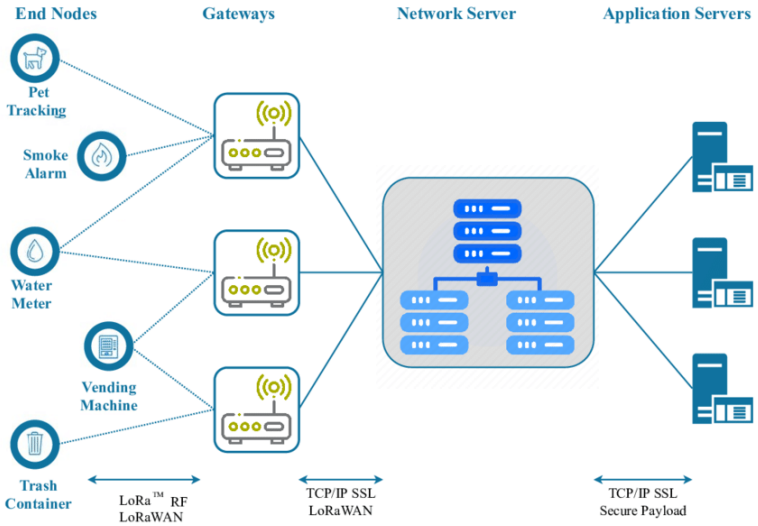

Figura 4.3: Arquitectura común de red LoRaWAN [22]

Como se ha mencionado, la bidireccionalidad es posible en la arquitectura de red, pero es más frecuente la comunicación *uplink* (desde los dispositivos finales hacia los servidores de red y aplicación) que los *downlink* (viceversa) debido a la naturaleza de las redes IoT.

4.2.2. Clases de LoRaWAN

El consumo de recursos y duración de la batería de los nodos de nuestra arquitectura de red LoRaWAN son esenciales y una de las ventajas más destacadas de esta tecnología. Por tanto, LoRaWAN define ciertos tipos de clases de comunicación entre nodos que afectará al consumo de batería de cada nodo. Estas comunicaciones son las que ocurren principalmente entre dispositivos finales y los gateways.

En general la comunicación LoRaWAN es ALOHA, es decir, es asíncrona y poco compleja. Generalmente, los nodos deben mandar transmitir únicamente cuando es requerido y es importante evitar tiempos excesivos de escucha y de despertar el nodo para ahorrar consumo de energía. Aun así, no todos los dispositivos LoRaWAN buscan funcionar de la misma manera y por ello se definen tres posibles comportamientos de comunicación con la red:

- **Dispositivos finales de clase A:** Representan la clase de dispositivo más común en LoRaWAN ya que es la que menos consumo de energía implica. Para cada comunicación desde el dispositivo final o nodo hacia la gateway, se permiten ciertas ventanas de recepeción para el *downlink*. La ventana está programada en función de lo que el propio dispositivo cree que necesita para recibir de vuelta información, además de un cierto tiempo aleatorio añadido para evitar colisiones. Es un claro uso de comunicación ALOHA, y se utiliza especialmente en dispositivos como sensores que la programación de comunicación no es muy compleja y donde la mayoría de las comunicaciones son en un único sentido.

- **Dispositivos finales de clase B:** Son dispositivos similares a los de la clase A, pero ofrecen ventanas planificadas de recepción *downlink* extras a las de la anterior clase. Los gateways envían a los dispositivos finales unas señalizaciones temporalmente sincronizadas, conocidas como *beacons*, que indican que deben abrir dichas ventanas de recepción. Es una manera efectiva para el servidor de controlar y saber cuándo los dispositivos finales están en escucha, pero tiene el defecto de consumir mayores recursos energéticos.

- **Dispositivos finales de clase C:** Al contrario que el resto de clases, este tipo de dispositivos se encontrarán escuchando continuamente sin interrupción salvo cuando son ellos los que están transmitiendo. Por una parte son muy útiles en dispositivos por el que hay un flujo constante de datos, pero el consumo de energía es bastante mayor. Posteriormente se comprenderá por qué esta es la clase de comunicación que es útil en este proyecto.

4.2.3. Autenticación y métodos de conexión

La seguridad no sólo es un aspecto añadido en nuestra arquitectura de red, sino que representa el método de conexión a la red desde cualquier dispositivo final. De esta manera, cualquier dispositivo que quiera conectarse a la red deberá de realizar un proceso de activación. Este proceso puede ser tanto por OTAA (*Over The Air Activation*) o ABP (*Activation By Personalization*).

Para comprender mejor estos dos métodos, es necesario enfatizar dónde se aplican estos procedimientos y qué parámetros utilizan. Comúnmente la seguridad en LoRaWAN se despliega en dos capas o entornos, la de red y la de aplicación. Garantizar la seguridad en la capa de red asegura la autenticidad de los dispositivos que se conectan, mientras que en la de aplicación se pretende que el operador red no disponga de los datos que se manejan en esta capa, normalmente los datos de aplicación del dispositivo final.

Por otra parte, los parámetros utilizados en el acceso a la red independientemente del tipo de procedimiento que se utilice son el *Application Identifier (AppEUI), Device Address (DevAddr), Network Session Key (NWSKey) y Applicaction Session Key (AppSKey)*. Estos dos últimos parámetros son claves que únicamente conocen tanto el dispositivo final como el servidor de red, pero su generación es diferente según qué método de seguridad se aplique:

- **OTAA:** Este método de activación es similar a un típico JoinRequest seguido de un *AcceptRequest*, donde el dispositivo final envía el servidor de red tres parámetros para la negociación de la autenticación, aparte de contar previamente ellos con una clave AES-128 llamada *AppKey*. Un esquema de comunicación que ayuda a comprender mejor este procedimiento es el de la figura 4.4 Un parámetros destacable (entre otros) que se utilizan en este método es el **DevEUI** que sirve como identificador global único del dispositivo.

- **ABP:** Para este procedimiento, no hay solicitud alguna y normalmente todos los parámetros utilizados son previamente definidos tanto en el dispositivo final como en el servidor de red.

En general, OTAA es un procedimiento más robusto y seguro ya que la creación de claves *NwkSKey* y *AppSKey* es dinámica y no son siempre las mismas, aunque por otra parte, el uso de ABP puede resultar más cómodo, especialmente en casos de práctica y simulación.

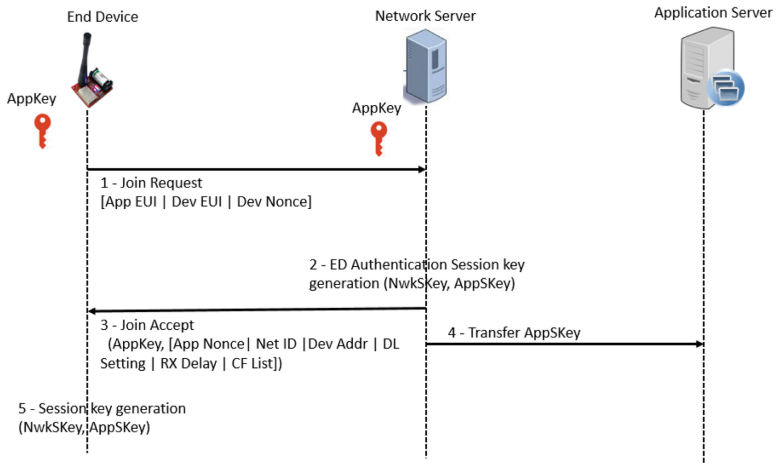

Figura 4.4: Esquema de comunicación para autenticación en OTAA [23]

4.2.4. Uso de ADR

Previamente se ha descrito la arquitectura de red y su topología, pero queda determinar los algoritmos para hacer posible que dicha arquitectura pueda ser realizada cumpliendo las características de optimización de recursos y codificación que se buscan.

Las especificaciones de LoRaWAN recomiendan el uso de ADR (Adaptative Data Rate) para maximizar el *throughput* de la red alterando la potencia de transmisión y la tasa de datos de los dispositivos finales (realmente se parametriza el SF, recordemos 4.2). Esta alteración de los parámetros de envío se basa en una estimación del link budget disponible (es decir, los recursos disponibles en el entorno de la transmisión) en el *uplink*, además de basarse en el SNR máximo para poder decodificar los paquetes a la tasa de datos existente. Esta estimación se realiza sobre experiencias previas de conexión de los dispositivos finales a la arquitectura de red. De esta manera, puede adaptar los parámetros de transmisión según las necesidades del dispositivo en concreto, pudiendo ahorrar recursos.

Esta adaptación a distintos dispositivos es posible gracias a que LoRa utiliza como técnica de modulación CSS y ésta ofrece ortogonalidad en las señales cuando se utilizan diferentes SF. Por tanto, los gateway hacen de transmisores-receptores con múltiples módem y múltiples canales, para poder recibir las diferentes señales con diferentes parametrizaciones desde distintos dispositivos finales.

ADR es clave en la colaboración de optimización de recursos en especificaciones LoRaWAN y la posibilidad de ser escalable, ya que no únicamente aprovecha las características de las técnicas de modulación LoRa para aho-

rrar consumo de energía y batería en los gateway (que a pesar de ser un elemento en nuestra arquitectura más potente que los nodos, recibe muchas conexiones) sino también el consumo en los dispositivos finales para poder adaptar su throughput y SF. Pero el uso de ADR no siempre está justificado, como en este proyecto donde no se utiliza porque se pretende mantener fijo el SF a 7, para garantizar la mayor velocidad posible.

4.2.5. Protocolo MQTT

Es necesario también abarcar y comprender el protocolo de mensajería más utilizado popularmente en redes IoT, y por tanto, también en LoRaWAN.

Se trata de MQTT, y es un protocolo de mensajería basado en publicaciones y suscripciones. De forma general, los dispositivos finales o clientes se suscriben a los temas que les interesan de los que el servidor publica a través de cierta entidad conocida como MQTT *broker*. Este MQTT *broker* es el componente central para permitir la comunicación entre clientes y dispositivos IoT, y recibe mensajes de los publicadores, verifican su autenticidad para después mandarlos a través de una cola en función de su QoS (*Quality of Service*) e identificar los suscriptores apropiados para recibir dicha publicación.

Este QoS tiene hasta tres niveles de elección [24] y define cuánto va a intentar de asegurarse que el mensaje sea recibido correctamente. Cada mensaje se puede publicar a cualquier nivel y el cliente puede intentar suscribirse con el mismo nivel u otro diferente. De esta manera, el cliente elige el nivel máximo de QoS que va a recibir pero sin poder elegir un nivel superior al del publicado. Estos tres niveles escalan como:

- QoS 0: At most once, es decir, el mensaje se va a enviar una única vez y no esperará confirmación.
- QoS 1: At least once, esta vez el mensaje se va a enviar al menos una vez y sí espera confirmación.
- QoS 2: Exactly once, se asegurará de que el mensaje llegue realizando un handshake de cuatro pasos.

Este protocolo tiene como característica ser poco costoso y consumir escasos recursos tanto de batería como de ancho de banda. Esto lo hace idóneo para entornos IoT pero también se utiliza por ejemplo en servidores cloud que manejan grandes infraestructuras y sirven como gran opción de ahorro en consumo energético.

Eclipse Mosquitto [25] es un ejemplo de MQTT *broker* de código abierto y es ampliamente utilizado en la comunidad científica para LoRaWAN e IoT en general. Es el software utilizado en este proyecto y viene ya incorporado en el servidor Chirpstack. Los mensajes MQTT vienen codificados en JSON/base64.

4.2.6. Arquitectura Chirpstack

Hasta el momento se han revisado ciertos elementos teóricos para conocer mejor el tipo de red que se quiere montar en este proyecto, pero es también importante conocer los principios de una de las herramientas utilizadas más imprescindibles en el diseño que posteriormente se describirá.

Chirpstack [12] es un repositorio de código libre que sirve como servidor de redes LoRaWAN. Fue desarrollado en 2016 por Orne Brocaar y anteriormente era conocido como LoRaServer. ChirpStack ofrece un entorno gráfico web para poder controlar y visualizar de forma más clara el funcionamiento y gestión de los dispositivos de la red. Una de sus características principales es que ofrece una API gRPC para permitir una comunicación entre diferentes dispositivos de forma óptima y pueda servir para integrar o extender Chirpstack, haciendo de esta una arquitectura modular.

Su arquitectura se puede definir como en la figura 4.5, donde la mayoría de componentes se conectan mediante procotolo MQTT, UDP y la API mediante gRPC. En este ejemplo se definen hasta cuatro gateways con diferentes conexiones al servidor Chirpstack, como son un *forwarder* de paquetes UDP, un *forwarder* para mensajería MQTT, combinación de tipos o establecimiento de conexiones básicas.

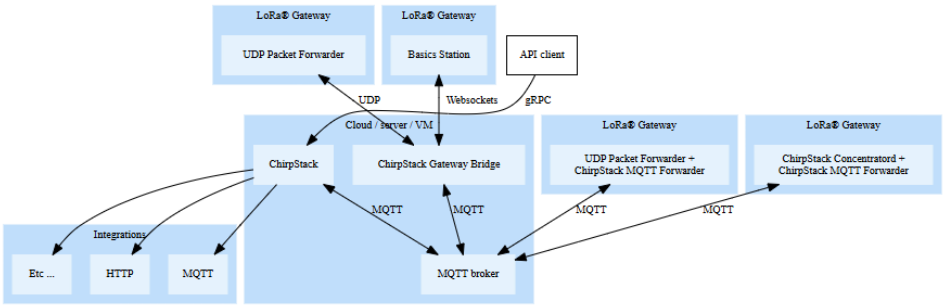

Figura 4.5: Arquitectura típica de Chirpstack con diferentes conexiones a las puertas de enlace

Por otra parte los componentes que se representan en este tipo de arquitectura clásica son los siguientes:

- Dispositivos LoRaWAN: Son la unidad básica en la arquitectura de red ya que representan los usuarios finales. Estos dispositivos pueden ser de varios tipos, pero todos tienen en común la capacidad de comunicación LoRa.
- Gateway LoRa: Sirven como puertas de enlace entre los dispositivos LoRaWAN y el servidor, en este caso Chirpstack, y especialmente como *packet forwarder*.
- Gateway Bridge de Chirpstack: Convierte los mensajes recibidos por gateways de tipo UPD o *websockets* en MQTT para la comunicación interna de Chirpstack y los publica en el MQTT *broker*.
- Chirpstack Concentratord: es un proceso *daemon* (que se encuentra en segundo plano) que se puede desplegar en las gateway para permitir una comunicación directa con el hardware del gateway. Este proceso se desarrolla como una API basada en ZeroMQ [26]. No se ha utilizado en este proyecto.
- Chirpstack: Anteriormente estaba dividido en servidor de red y servidor de aplicación, pero recientemente se unificaron en un sólo componente. Aparte de ofrecer las anteriores prestaciones propias de un servidor de red como control y gestión de la red, conexiones y autenticación, también despliega una interfaz web gráfica para el control de gateways y dispositivos y desencripta los *payloads* de las aplicaciones recibidas.

Cabe mencionar que Chirpstack también gestiona los algoritmos y funcionalidades propias de LoRaWAN, como las autenticaciones OTAA y ABP anteriormente repasadas. Incluye posibilidad de ADR, que debe ser solicitada por el dispositivo LoRaWAN para que el servidor ajuste su tasa de datos y potencia de transmisión, y es válido tanto con toda clase diferente de dispositivo (clase A, B, C).

Generalmente es uno de los servidores de mayor popularidad en arquitecturas LoRaWAN por su simplicidad de uso, funcionamiento y despliegue.

4.2.7. Limitaciones generales

Una limitación general en una red LoRaWAN es la del acceso al medio. Como se ha tratado anteriormente LoRaWAN utiliza un mecanismo ALOHA, el cual hace este mismo proceso muy sencillo pero con posibilidad de colisiones cuantos más dispositivos dispuestos a comunicar haya en la red. A pesar de que se aplican ciertas medidas para evitar colisiones (como aleatorizar un tiempo en el envío desde los dispositivos finales), es necesario tener en cuenta que este tipo de pérdidas y colisiones pueden ocurrir.

Por otra parte, las capacidades de transmisión de TCP/IP y LoRa no son las mismas. En nuestro contexto, que se planea habilitar un punto de acceso IP para dispositivos LoRa, se debe comprender la limitación que existe para enviar información. Donde en TCP/IP puede ser común el envío de audio y vídeo, en LoRaWAN no es práctico hacer uso de su arquitectura de red para este tipo de contenidos. Es necesario determinar ciertos factores en LoRaWAN para poder conocer si es viable o no el envío de información y cómo. Por ejemplo, el uso de bandas ISM comentado anteriormente en Europa se encuentra limitado a 10.94 kbps para un ancho de banda de 250 kHz. Pero la tasa de datos está limitada al SF utilizado y en ciertos contextos donde se quieren aprovechar al máximo las características fuertes de LoRaWAN, la tasa de datos será más baja.

Estas limitaciones son la razón principal por las que en este proyecto no se pretende utilizar aplicaciones que supongan un envío de información mucho más grande del que estos sistemas pueden realizar. A cambio, el objetivo es ser capaz de poder enviar información pequeña pero valiosa como una ubicación GPS, determinados mensajes o similares, que sí se pueden admitir en una arquitectura de red LoRaWAN y aprovechar sus características.

Capítulo 5

Diseño e Implementación

En este capítulo se presenta la elección de diseño para conformar el objetivo del proyecto, cómo se ha implementado y sus características. Primero se detallará el flujo de comunicación existente en la arquitectura de red del proyecto para poder mandar paquetes IP a través de LoRaWAN. Posteriormente se analizarán los componentes principales de la arquitectura de red en cada sección, las aplicaciones utilizadas y los archivos de configuración necesarios para poder hacer funcionar dicho flujo de comunicación.

5.1. Diseño del flujo de comunicación

5.1.1. Uplink

El flujo de comunicación empieza en el *uplink* mediante el envío limitado de paquetes IP a través de un CPE, en este caso un dispositivo móvil conectado a una mota que sirve como dispositivo LoRaWAN. Esta conexión se crea gracias a la posibilidad que tiene la mota de desplegar un punto de acceso Wifi, donde el CPE envía los paquetes deseados. La mota fragmenta los paquetes IP recibidos y los reenvía mediante LoRaWAN al servidor pasando antes por el dispositivo gateway asociado.

Una vez llega al servidor, los payloads en JSON/base64 que son los fragmentos del paquete IP son procesados normalmente por el servidor de aplicación y llegan al MQTT *broker* en un *topic*. Es el *forwarder* el que se suscribe al *topic* del MQTT *broker* para rescatar los paquetes enviados y reconstruirlos nuevamente utilizando la librería Scapy [27] en Python. Es también responsable de cambiar la IP origen por la IP del host Ethernet (SNAT) para posteriormente reenviarlo al destino de la red local.

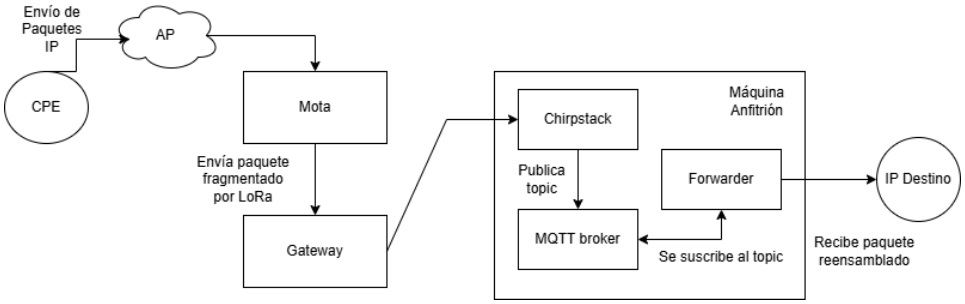

Figura 5.1: Flujo de comunicación en el uplink y compontes involucrados

5.1.2. Downlink

Una vez son recibidos los paquetes IP por el destinatario, éste responde con un paquete IP estándar porque está entendiendo en todo momento que está recibiendo información de una IP válida en LAN. De esta manera, esta respuesta la capta nuevamente el *forwarder* gracias a que habilita un *sniffer ethernet* en la interfaz de destino, y vuelve a cambiar la IP destino de los paquetes de respuesta por la original de la mota, fragmentar los paquetes para poder enviarlos por LoRaWAN y reenviarlos ahora a través de la API gRPC de Chirpstack.

Por último, la mota recibe los paquetes y vuelve a re-ensamblar para reenviarlos a la dirección destino.

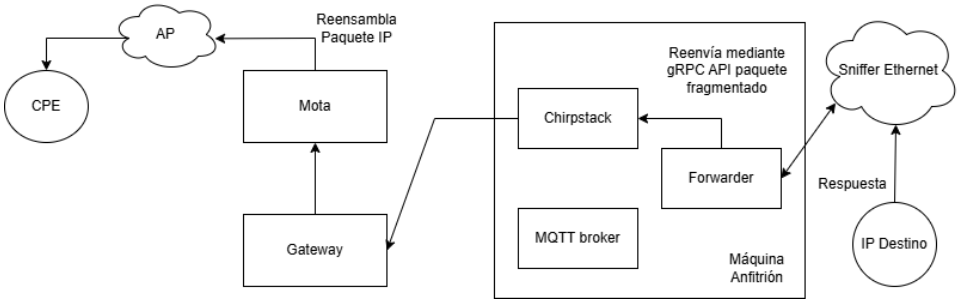

Figura 5.2: Flujo de comunicación en el downlink y componentes involucrados

5.2. Servidor

La configuración del servidor Chirpstack es bastante estándar, así como su instalación (véase A). Lo más importante es configurar en la plataforma Chirpstack nuestra arquitectura de red para que conozca el dispositivo LoRaWAN que se va a utilizar, el *gateway* y la aplicación.

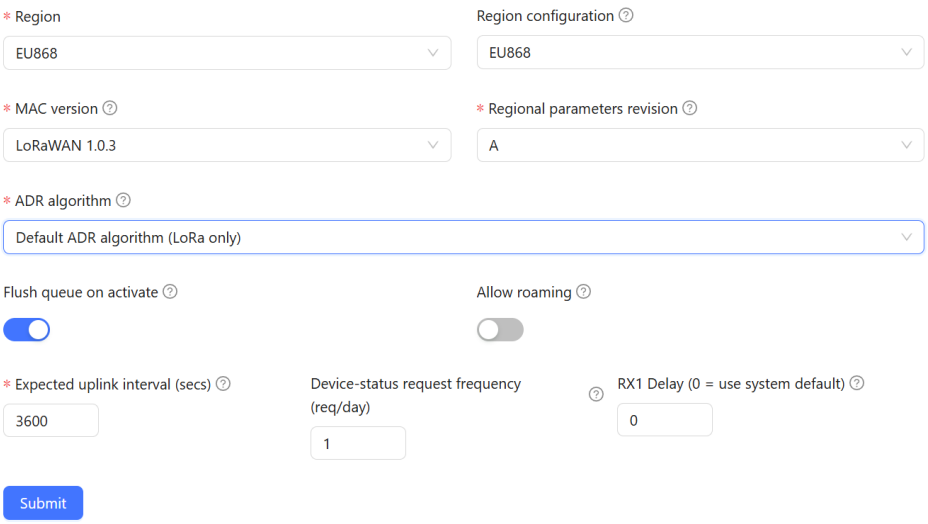

Figura 5.3: Configuración básica para la creación de un perfil

Hay ciertos parámetros a configurar en Chirpstack para nuestro diseño. Por una parte en la figura 5.3 se configuran aspectos básicos del perfil creado, como la región EU868, la versión MAC de LoRaWAN que soporta nuestro dispositivo, el algoritmo ADR y que limpie la cola por cada activación de OTAA y ABP (clave para hacer pruebas y no acumular *uplinks* anteriores).

Otra configuración importante es indicar que se va a utiliza OTAA (en el foro de Chirpstack apuntan que utilizar ABP puede dar problemas al tratar al dispositivo como uno de clase A [28]) e indicar el uso de un dispositivo de clase C con un *downlink* esperable de cada 0 segundos. Esto último se hace por que se pretende ser capaz de recibir y escuchar en el *downlink* en la mayoría de casos que se esté utilizando el dispositivo. Hay que tener en cuenta que en caso de pruebas de tráfico ICMP y ejecutar varios *ping* es importante estar escuchando la respuesta en todo momento, pero también es aplicable a TCP o UDP.

Por último, a pesar de que en la interfaz de Chirpstack se indique el uso de ADR, realmente se debe configurar en el archivo *region_eu868.toml* un *data rate* de valor 5, que es el valor de tasa de transmisión utilizado en el diseño del proyecto porque es el que permite una mayor velocidad de transmisión. Esto podría ser cambiado a otro valor distinto para manejar otra velocidad de transmisión a cambio de mejor sensibilidad en la recepción de la señal, pero también se debería de editar de la misma manera en el resto de componentes del proyecto.

5.3. Forwarder

Este componente es esencial en el diseño del proyecto tal y como se ha descrito en el flujo de comunicación. Es un código de Python que se ubica en la máquina anfitriona del servidor LoRaWAN y se encarga tanto de seleccionar los paquetes fragmentados que llegan al MQTT *broker* y ensamblarlos para ser enviados mediante Ethernet al destinatario, como de fragmentar la respuesta y volverla a mandar por la red LoRaWAN. La ejecución del código requiere de parámetros de entrada que se especifican como variables en el inicio del código, como son:

- Parámetros relacionados a la API: APISERVER y APITOKEN para poder establecer conexión directa con el servidor Chirpstack, siendo el primero la dirección IP del servidor y el segundo el *token* generado desde Chirpstack para poder habilitar la API.

- Parámetros relativos a MQTT: MQTTUSER y MQTTPASS para establecer usuario y contraseña en la sesión MQTT, MQTTPORT para especificar el puerto MQTT que normalmente es 8883 en caso de TLS/SSL y 1883 en caso opuesto, MQTTSERVER que es la dirección IP donde se ubica el MQTT *broker*, los tópicos TOPIC_RX y TOPIC_TX a lo que el forwarder se debe suscribir tanto *uplink* como *downlink* y por último el certificado ca.crt necesario en conexión TLS/SSL.

- Parámetros relacionados al reenvío: como el parámetro INTERFACE que indica la interfaz de red a la que el *forwarder* va a reenviar los paquetes IP para que sean transportados por la red local, y el parámetro DEVEUI que define el identificador del dispositivo final LoRaWAN, al que debe indicar al servidor Chirpstack que realice la comunicación *downlink*.

5.3.1. Fragmentación de Paquetes IP

A continuación se va a repasar las distintas funciones del código *forwarder*. Empezando por las relacionadas a la fragmentación de paquetes se encuentran dos funciones:

- Una propia para la **fragmentación de paquetes IP** en el *downlink*. Esta función utiliza principalmente la librería Scapy para la manipulación de paquetes IP en Python y poder extraer de ellos el encabezado e identificador del paquete IP para poder ser utilizado como identificador del fragmento.

Fragmenta el *payload* del paquete original en divisiones suficientes como para que entren en fragmentos de unos 215 bytes (pensado para SF7) incluyendo un encabezado especial de 6 bytes que incluye el número de secuencia, el ID del paquete, si existen más fragmentos aún y una *Magic Word* de 2 bytes que sirve como identificador en la arquitectura de red. Además el primer fragmento incluye, aparte de estos campos, el encabezado original IP. Un ejemplo de fragmentación de un paquete es:

1. Fragmento 0: Encabezado(6 bytes) + cabeceras IP + primeros bytes del *payload*.
2. Fragmento 1: Encabezado(6 bytes) + siguiente parte del *payload*.
3. Fragmento 2: Encabezado(6 bytes) + siguiente parte del *payload*.
4. Fragmento 3: Encabezado(6 bytes) (donde se indica que no hay más fragmentos restantes) + última parte del *payload*.

- **Cola de los fragmentos creados**: Almacena temporalmente los fragmentos que salen de la anterior función para poder temporizar el envío de forma adecuada. Está diseñada para no poder tener más de diez fragmentos en cola y la función de fragmentación de paquetes no le enviará más fragmentos hasta que la cola tenga hueco nuevamente.

5.3.2. SNAT

Además de la fragmentación de paquetes IP se incluyen unas clases relacionadas a una de las operaciones del *forwarder* que es el SNAT. Este SNAT es necesario aplicarlo para cambiar la dirección de la IP origen por la de la dirección del *gateway* para poder tener una dirección IP válida que nuestro servidor LoRaWAN conozca. Aplicar SNAT es tan fácil como modificar parámetros mediante Scapy, pero es necesario además almacenar un mapeo de las rutas para poder recordar qué dirección IP o puerto fue cambiado. Esto se hace con las clases:

- **PortMapping**: para TCP/UDP, almacena las direcciones y puertos originales y se reinicia si un nuevo paquete que coincide con el puerto anterior llega de nuevo. Por otra parte tiene un tiempo límite para expirar de 300 segundos.

- **ICMPMapping**: funciona de manera similar que el anterior pero para ICMP, de manera que no almacena los puertos utilizados si no un el identificador ICMP como el de un *ping*.

5.3.3. Funciones asociadas a MQTT

La comunicación del *forwarder* con el MQTT *broker* es importante en todo momento y por tanto se debe establecer funciones básicas para establecer una suscripción al *topic* que previamente se había especificado en los parámetros de entrada. Esto es posible gracias al uso de la librería *paho-mqtt* [29] de Eclipse, y se realiza en dos funciones diferentes, una de suscripción al *topic* y otra que ejecuta esta misma, aportando el certificado de seguridad, usuario y contraseña pertinentes. Por último, la función **on_message** sirve para indicar qué hacer con los fragmentos recopilados de la suscripción.

La función **on_message** define el comportamiento del *forwarder* respecto a los fragmentos que recibe desde el MQTT *broker*. Este comportamiento se puede dividir en dos fases:

- Re-ensamblar fragmentos y formar el paquete IP: Empieza haciendo un *parse* de JSON para convertir la cadena de texto en un objeto estructurado y extrae y decodifica los datos en base64. Por otra parte, verifica si es un paquete de la red LoRaWAN con la palabra mágica, y si todos los fragmentos han sido recibidos (comprobándolo cuando llega el fragmento con el último número de secuencia y si todos los números de la secuencia están disponibles, si no, se queda esperando) se construye el paquete IP con las direcciones de destino y de origen originales.
- Ajustar el paquete IP para el reenvío: Una vez todos los fragmentos llegan de forma exitosa y se ha formado el paquete IP, se construye otro similar que servirá de copia para en este hacer el SNAT y cambiar la dirección de origen. Esto implica también re-calcular los *checksums* del paquete IP (verificar que en la modificación de paquete no ha sido alterado) y sus protocolos, y según qué protocolo se utilice (ICMP, TCP o UDP), se hará un mapeo u otro. Por último, el paquete es reenviado con Scapy por la interfaz de red indicada.

5.3.4. Funciones asociadas a gRPC API

La plataforma Chirpstack hace la mayoría de trabajo en la comunicación *downlink* mediante su gRPC API. El *forwarder* lo único que tiene que construir es la solicitud de *downlink* adjuntando los parámetros de entrada que ya habían sido indicados, como la dirección del servidor API, el *token* y el identificador del dispositivo LoRaWAN. Una vez construida esta solicitud, simplemente se la manda a Chirpstack para que realice el reenvío de información. Todo esto está indicado en la función **EnqueueDownlink**.

El tipo de solicitud en la arquitectura API de Chirpstack es de tipo

EnqueueDeviceQueueItemRequest [30], utilizado para mandar un mensaje a la cola de transmisión hacia un dispositivo LoRaWAN. De esta comunicación además, no se espera ningún ACK o mensaje de respuesta.

5.3.5. Sniffer Ethernet

El *sniffer Ethernet* es uno de los componentes más importantes en la comunicación *downlink* de la arquitectura de red del proyecto. Gracias a este componente es posible rescatar el mensaje de respuesta del destinatario y que el *forwarder* sea capaz de reenviarlo a través de la red Chirpstack. Para formar el *Sniffer Ethernet* se utiliza nuevamente la librería Scapy, utilizando *sniff()* sobre la interfaz Ethernet elegida que es donde llegará la respuesta de la máquina destino en el *downlink*.

Primero determina si el paquete es TCP, UDP o ICMP para invertir de vuelta la dirección original del paquete que anteriormente había sido cambiada (SNAT) en el *uplink*. Por eso determina el protocolo utilizado para saber cuál mapeo utilizar y poder restaurar su dirección original en un paquete clonado al que se ha interceptado.

Una vez hecho esto se re-calculan los *checksum*s nuevamente, se fragmenta el paquete IP y se reenvía por el gRPC API de Chirpstack.

5.4. Dispositivo LoRaWAN

El dispositivo LoRaWAN o la mota en nuestro proyecto como se ha mencionado anteriormente es un *Heltec Wireless Stick V3*, y se ha programado con Arduino IDE. La librería utilizada es Heltec ESP32 Dev-Boards 2.1.2, y el gestor de las placas *arduino* en Arduino IDE es ESP32 v3.0.7 por Espressif Systems.

Este dispositivo junto al *forwarder* son los dos responsables de manejar y manipular los paquetes IP en la arquitectura de red. Anteriormente, se ha visto cómo el forwarder tiene como objetivo hacer llegar y capturar los paquetes de envío y respuesta al destinatario. En este caso, el dispositivo LoRaWAN tiene como objetivo cumplir lo mismo pero con el usuario emisor.

5.4.1. Creación del AP

El punto de acceso (AP) Wifi es la primera herramienta que nuestro dispositivo LoRaWAN crea para ser capaz de recibir información directa de un CPE cercano. Desde la perspectiva de un usuario final, este punto de acceso wifi es lo único que ve de forma directa en toda la arquitectura de red.

Este punto de acceso se puede crear gracias las capacidades del dispositivo LoRaWAN escogido por la librería *esp_wifi.h* [31]. Se puede establecer el SSID del punto de acceso (el identificador o nombre con el que aparecerá para el resto de usuarios), su contraseña, tipo de autenticación (en este caso WPA2) y dirección DNS para poder ofrecer un servicio de DHCP para asignar direcciones IP a los dispositivos que se unan a la red.

Por otra parte el MTU (cantidad máxima de bytes que se puede enviar a través de una interfaz de red) está definido a 1500, que es lo habitual. En nuestra red nos interesa fragmentar los paquetes para poder ser enviados sin problemas a través de LoRaWAN, pero se pretende que sea el propio dispositivo LoRaWAN que haga esta fragmentación una vez obtiene los paquetes IP de forma directa desde el usuario final en vez de que lo haga cada usuario imponiendo una MTU más pequeña. Además, realizándolo de esta manera se asegura un escenario mucho más controlado ya que la gestión de la fragmentación IP está centralizada.

5.4.2. Funciones Wifi

Son todas las funciones necesarias que utiliza el dispositivo LoRaWAN para poder manejar el tráfico recibido. Esto incluye la definición de un *firewall* para catalogar el tráfico recibido al punto de acceso, su procesamiento y su reenvío.

- **Firewall:** Su objetivo es reconocer los paquetes que llegan al dispositivo LoRaWAN. De forma general, el tráfico que llega al punto de acceso se manda a la pila de protocolos LWIP (*Light-Weighted IP*, que sirve generalmente para redes ligeras o sistemas embebidos) para poder procesar los paquetes ARP o DHCP necesarios pero para también distinguir y poder reenviar los paquetes Ethernet del protocolo utilizado (ICMP, TCP o UDP) en caso de que el destinatario sea la red local. También es capaz de reconocer IPv6 pero su implementación en este sistema no existe, por ello, también se mandan a la pila IWIP.

  Los únicos paquetes que no se mandan a esta pila de protocolos son los de ICMP, TCP o UDP que se quieren enviar a través de la red LoRaWAN en este proyecto.

- **Funciones de fragmentación:** La lógica e implementación de las funciones de fragmentación es muy similar a la del *downlink* de la sección 5.3.1. De nuevo hace uso de una generación de una cola o *buffer* para colocar los fragmentos de paquete a enviar por la red. Además, como es lógico, sigue el mismo diseño de fragmentación de *payload* acompañado de un encabezado especial que contiene la palabra mágica, número de secuencia e ID del paquete.

La diferencia más notable es la adaptación de lenguajes de programación y entornos, puesto que uno está implementado en Python con uso de la librería Scapy y otro en C++/C con la librería ESP de Espressif Systems.

- **Gestor de eventos:** Aunque su implementación no tiene un gran impacto directo en el funcionamiento del dispositivo LoRaWAN ni la red, sí es importante ya que sirve como una función *debug* de los eventos que van ocurriendo en el punto de acceso. Indica el inicio o desconexión del punto de acceso, registra las direcciones IP conectadas, si un paquete pasa por el *firewall* o la dirección asignada por DHCP a un usuario.

- **ARP:** Es imprescindible la creación de una tabla ARP que sirva como traductor de direcciones MAC a IP. Esto se implementa de forma manual en el dispositivo LoRaWAN mediante la creación de una tabla con un número limitado de entradas. Existen dos funciones, una para añadir o actualizar una entrada nueva a la tabla ARP con la dirección IP y su dirección MAC correspondiente, y otra función para traducir una dirección IP que se encuentre en la tabla a dirección MAC de forma directa.

5.4.3. Configuración LoRaWAN

Estas funciones son las comunes en un dispositivo LoRaWAN que forma parte en dicha arquitectura, aunque en este diseño e implementación hay ciertas características que son interesantes comentar.

De forma general, dispone una función parecido a una máquina de estados propia de un dispositivo LoRaWAN. Esta incluye estados para el inicio del *stack* LoRaWAN y sus parámetros, conexión a la red (sección 4.2.3), envío de datos, programación del siguiente ciclo o entrada en bajo consumo, además de indicar cuándo se ha completado una transmisión. El *uplink* hace uso de la mayoría de estados y además escoge los paquetes que se encuentren disponibles en la cola de *uplink* donde se habían ido guardando los fragmentos IP para enviar.

Por otra parte el *downlink* en este sistema es algo más complejo pero similar al del *forwarder*. La característica principal de este sistema es que el dispositivo LoRaWAN utilizado está preparado para ser utilizado como un dispositivo de clase C (sección 4.2.2) y por tanto define dos ventanas de recepción para el *downlink* conocidas como RXWIN1 y RXWIN2. Como la comunicación *downlink* en nuestra arquitectura de red está realizada a través de la gRPC API de Chirpstack, se debe utilizar esta segunda ventana, que se abre dos segundos después del *uplink* con un *Data Rate* definido por el

servidor de 5 (equivalente al SF 7 en la región EU868). Una vez definida la ventana de recepción en el *downlink*, re-ensamblar los paquetes IP es también similar a cómo los haría el *forwarder* en la sección 5.3.3 que sería comprobando la palabra mágica, extrayendo los parámetros del encabezado y reconstruyendo el paquete original.

5.5. Resto de configuración

Otras configuraciones que pueden ser necesarias en esta red es la implementación de algún tipo de cortafuegos que limite el tráfico externo del CPE para evitar saturar el dispositivo LoRaWAN y así poder aislar el entorno de pruebas. A pesar de que esto ya está indirectamente implementado con la función *firewall* del dispositivo LoRaWAN (sólo acepta el tráfico ICMP, TCP y UDP), añadir otro sistema que filtre posible tráfico molesto puede ser útil.

En caso de utilizar un ordenador, se pueden añadir *iptables* para únicamente permitir el tipo de tráfico que nos interese con los protocolos que nos interesan y puertos necesarios. En caso de utilizar un móvil se puede utilizar cualquier tipo de aplicación gratuita. En este proyecto se utiliza *NetGuard* tal y como se describe en el B porque no requiere de permisos de superusuario, pero cualquier otro tipo de aplicación que limite el tráfico externo del dispositivo móvil serviría.

Por último, en caso de utilizar protocolo TCP, es importante añadir la siguiente regla de *iptables* en la máquina anfitriona del *forwarder*:

```
iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP
```

De esta manera, se evita que la máquina reinicie las conexiones TCP porque descarta cualquier paquete con el flag *RST*.

Capítulo 6

Evaluación de la Implementación

En este apartado se realizarán y evaluarán pruebas sobre el diseño y la implementación presentados, con el objetivo de determinar las capacidades de la red propuesta y obtener resultados concluyentes. Toda la instalación de la red y cómo se lanza se encuentra en los anexos A y B, que servirán de base para la realización de las pruebas.

Como se ha repasado, el diseño propuesto admite tres tipos de protocolos: ICMP, UDP y TCP. La elección de estos tres protocolos es bastante intuitiva, puesto que ICMP al ser un protocolo de control presenta un gran escenario para hacer pruebas de latencia sobre la red, mientras que UDP y TCP pretenden servir escenarios reales de intercambio de información a modo chat mediante LoRaWAN. En general se pretende exponer el funcionamiento básico de cada uno de los protocolos y para ICMP en concreto, realizar pruebas de latencia para mostrar las capacidades de la red.

Por otro lado, el escenario físico donde se han ejecutado estas pruebas es en un entorno de laboratorio. Ciertos factores como la distancia entre dispositivo LoRaWAN y puerta de enlace, presencia de otro tipo de dispositivo LoRaWAN, o establecer el lugar físico en una habitación puede afectar al rendimiento de la red. Se debe recordar que esta red está pensada para colocarse en zonas sin cobertura móvil (normalmente zonas rurales o montañosas) donde colisiones de otro tipo de dispositivos son menos probables y la presencia de menos obstáculos favorecen las comunicaciones. Igualmente, para realizar pruebas de funcionamiento básicas, el escenario planteado es perfectamente funcional y válido.

Para comprender mejor el escenario de red utilizado y los posteriores resultados, se detallan las direcciones IP de los dispositivos en el cuadro 6.1.

Máquinas	Dirección IP
CPE	192.168.4.2
Dispositivo LoRaWAN	192.168.4.1
Máquina objetivo	150.214.190.85
CPE tras SNAT	192.168.123.108

Cuadro 6.1: Direcciones IP relevantes

A continuación se dividirá la evaluación de las pruebas realizadas sobre cada uno de los protocolos y se presentarán qué escenarios se han planteado para cada uno.

6.1. ICMP

En este protocolo es el que más pruebas se han realizado gracias a su escasa complejidad y fácil obtención de datos. En las siguientes pruebas se pretende obtener el *delay* o latencia en milisegundos mediante *pings* para conocer la velocidad de la red frente a diferentes *payloads*.

El CPE hará *pings* a una máquina conocida en la red del *forwarder*, en concreto, se ejecutarán aproximadamente unos 25-50 *pings* (dependiendo del *payload*) con un intervalo de tiempo suficiente para no saturar la cola de nuestro dispositivo LoRaWAN. Este intervalo se debe ajustar al tiempo que tarda nuestro dispositivo LoRaWAN en recibir cada uno de los fragmentos en caso de que el paquete IP haya sido fragmentado. Es decir, si se esperan unos 3 fragmentos que suelen tardar de media unos 2 segundos, es recomendable ajustar un intervalo de *ping* de 10 segundos para garantizar que no haya cuello de botella. Es importante tener esto en cuenta porque de esta red se espera una recepción mucho más lenta de lo habitual comparado a otro tipo de redes, y por ello, para ver la velocidad real de la red es aconsejable también aumentar el intervalo de envío.

6.1.1. Sin fragmentación

La primera prueba se realizará con un tamaño de paquete estándar cuando se realiza un *ping* en Linux, que es de 84 bytes en total (56 bytes de datos, 8 de cabecera ICMP, y 20 de cabeceras IP y después se ha de contar con 6 bytes más que se han añadido en el diseño de la red como cabeceras especiales). Esta primera prueba servirá como muestra inicial del funcionamiento de la red.

Se han realizado en total cerca de 60 *pings*, de los cuales han tenido éxito 37 que son los que se van a considerar. Nótese que a pesar de sufrir un medio-alto porcentaje de pérdidas, este no es el objetivo final a evaluar en este

proyecto. Además, muchas de estas pérdidas se pueden deber a colisiones y factores externos de haber realizado las pruebas en un entorno de laboratorio como ya se ha mencionado.

Como la red está diseñada para fragmentar datos a partir de 221 bytes, para este tamaño aún no se hará fragmentación pero igualmente es recomendable especificar un intervalo de 10 segundos para evitar llenar las colas de envío y recepción.

En el CPE entonces se ejecuta: *ping -c 60 -i 10 150.214.190.85*

Antes de revelar los valores de latencia, se comprueban los *logs* de los distintos componentes de la arquitectura de red.

En los *logs* de MQTT, por cada *ping* aparecen tres tipos de eventos que son de *up (uplink)*, *txack (downlink)* y *log* (este último no es relevante porque lo único que indica es que el payload recibido no es el esperado, pero no afecta a la implementación de red y es algo que ya se conocía). La mayoría de la información que proporcionan estos eventos son las características de la comunicación LoRaWAN, desde qué dispositivo LoRaWAN se ha utilizado, *gateway*, localización, características de la transmisión (*rssi, snr, canal y frecuencia utilizados, data rate...)*, hora del evento, de qué aplicación proviene y los datos enviados entre otros (similar a la figura A.2). Es una buena señal de que la arquitectura funciona tanto en uplink como downlink y que nuestra aplicación es funcional y los dispositivos de la arquitectura tanto el dispositivo LoRaWAN como el gateway funcionan.

En el *log* del *forwarder* (figura 6.1) también se comprueba que éste cumple su función. Al estar suscrito al tópico de la aplicación correspondiente, recibe y re-ensambla el paquete IP (*echo-request)* proveniente del CPE. Se guarda la dirección original del emisor y aplica SNAT para darle una dirección válida que el destino sí conoce y se lo envía a través de la interfaz especificada. Ya en el *downlink*, con el *sniffer Ethernet* intercepta el paquete con la misma dirección destino que la dirección a la que se había cambiado con SNAT, lo fragmenta de nuevo y lo reenvía a través de la API de Chirpstack con la dirección original. Como no hace fragmentación la red en ningún momento, únicamente recibe un paquete en la ida y envía otro en la vuelta, de tamaño 90 bytes.

```
>>> MQTT message received (UL) on topic 'application/0d812802-5163-4851-8ec6-7de79b842cd7/device/70b3d57ed00653c9/event/up'
2025-06-11 09:27:42,312 - INFO - Received IP packet summary: IP / ICMP 192.168.4.2 > 150.214.190.85 echo-request 0 / Raw
2025-06-11 09:27:42,315 - INFO - Constructed IP packet summary: IP / ICMP 192.168.123.108 > 150.214.190.85 echo-request 0 / Raw
2025-06-11 09:27:42,315 - INFO - Constructed IP packet (hex): 450005d643e940003f0160fdc0a87b6c96d6be550800ad6003e80001ee4b496800000000ec180c000000000
2025-06-11 09:27:42,316 - INFO - ==> Added ICMP mapping: ID=1000 to src IP 192.168.4.2
2025-06-11 09:27:42,316 - INFO - Valid IP packet with src IP 192.168.4.2 (ICMP, ID=1000, seq=1) -> changing src IP to 192.168.123.108 (source NAT),
forwarding it via network interface enp86s0...
2025-06-11 09:27:42,377 - INFO -
<<< Received packet (DL) matches ICMP mapping: 192.168.123.108 (ID: 1000)
2025-06-11 09:27:42,378 - INFO - Received IP packet summary: Ether / IP / ICMP 150.214.190.85 > 192.168.123.108 echo-reply 0 / Raw
2025-06-11 09:27:42,382 - INFO - Constructed IP packet summary: IP / ICMP 150.214.190.85 > 192.168.4.2 echo-reply 0 / Raw
2025-06-11 09:27:42,383 - INFO - IP datagram fragmented successfully.
a93372d0-a9b4-433e-be10-90cb47975042
2025-06-11 09:27:42,408 - INFO - Sent message via gRPC API with fragment. Length: 90
2025-06-11 09:27:42,408 - INFO - Sent fragment: 0x55aa29c00001450005d629c000004001319196d6be55c0a804020000b56003e80001ee4b496800000000ec180c000000000
5a446865-6277-4527-8bfe-7dd9a8c1edca
```

Figura 6.1: *Log* del *forwarder* para *ping* de tamaño 90 bytes

```
joining...
[INFO] LoRaWAN frame sent
joined
[INFO] Sending first LoRaWAN frame so we can receive in downlink...
[INFO] LoRaWAN frame sent
received unconfirmed downlink: rssi = -87, snr = 0, datarate = 5
[INFO] LoRaWAN frame sent
received unconfirmed downlink: rssi = -78, snr = 8, datarate = 5
[INFO] Received downlink frame: RXWIN1, size 66, port 1
       REV DATA: 0x55AA3BBB00004500003C3BBB00004001253096D6BE55C0A80
45 00 00 3C 3B BB 00 00 40 01 25 30 96 D6 BE 55
C0 A8 04 02 00 00 55 28 00 01 00 33 61 62 63 64
65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74
75 76 77 61 62 63 64 65 66 67 68 69
88 F4 DA E4 AC 3C 4A CA 43 56 21 F8 08 00 45 00
00 3C 3B BB 00 00 40 01 25 30 96 D6 BE 55 C0 A8
04 02 00 00 55 28 00 01 00 33 61 62 63 64 65 66
67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76
77 61 62 63 64 65 66 67 68 69
```

Figura 6.2: Serial monitor de Arduino IDE para dispositivo LoRaWAN al envío y recepción del *echo-request* y *echo-reply*

Por último, en el *log* del dispositivo LoRaWAN (figura 6.2) o lo que se ve en el *Serial Monitor* de Arduino IDE, también se ve el envío de la trama LoRaWAN (tras haber formado el paquete IP con las cabeceras especiales) y la recepción posterior en el *downlink* del paquete. Precisamente, si pasamos los datos recibidos en hexadecimal del paquete IP (ya sin cabecera especial) por un decodificador online [32], se puede comprobar la información de las cabeceras IP e ICMP (figura 6.3). Se puede ver que es de tipo *Echo (ping) reply* tal y como se esperaba, y que la dirección destino es el CPE.

En la figura 6.4 se encuentra un histograma con los valores obtenidos del tiempo de respuesta de de los *pings*, así como una curva con la densidad de probabilidad estimada (línea naranja). El histograma está configurado para que siga una tamaño de *bin* según el método densidad de frecuencia en Matlab. Aparte, el cálculo de la distribución *log-normal* permite conocer mejor los datos que se han obtenido:

- Media aritmética (ms): 1565.15
- Desviación estándar (ms): 50.80

Son unos valores claramente mayores a los que se encuentran en otro tipo de arquitecturas red más comunes para IP. Pero la primera conclusión que se puede obtener es que la mayoría de los valores son bastante estables y el tiempo de respuesta de la red no sufre grandes alteraciones entre intentos de *ping*. Seguramente el mayor efecto de retardo en la red debe ser el tiempo en

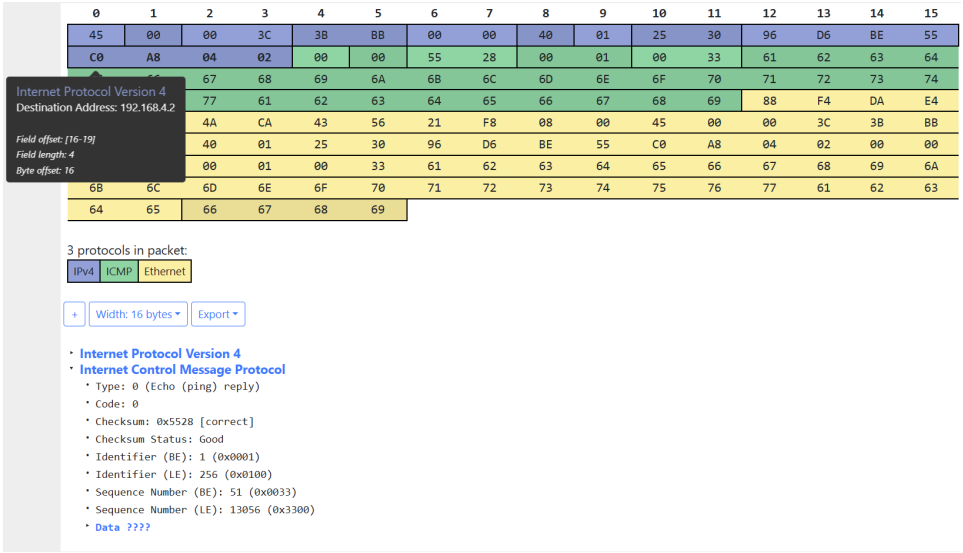

Figura 6.3: Paquete IP en el *downlink* decodificado

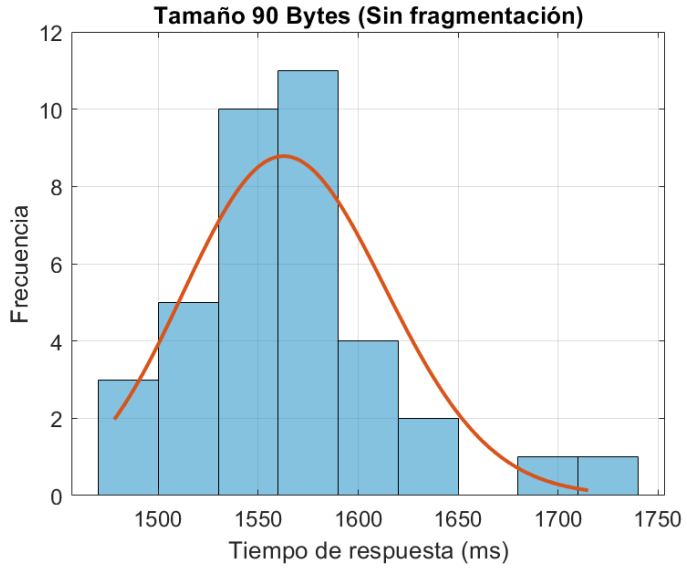

Figura 6.4: Histograma (azul) de los *pings* realizados para tamaño 90 bytes y densidad de probabilidad estimada (naranja)

el aire de envío del paquete por LoRaWAN y es también el que quizá más variaciones pueda sufrir pero en este escenario se mantiene generalmente estable.

A continuación, se incrementará el valor de envío para ver qué diferencia en el tiempo de respuesta ocurre pero aún evitando la fragmentación de paquetes. Un tamaño apropiado para la ejecución de esta prueba puede ser el de 200 bytes (ya incluyendo cabeceras especiales del diseño), por lo tanto la configuración de *ping* es: *ping -s 166 -c 50 -i 10 150.214.190.85* , donde se realizan 50 *pings* de tamaño 194 bytes (166 de datos más 8 de cabeceras ICMP más 20 de IP) que serán 200 bytes cuando el dispositivo LoRaWAN añada las cabeceras especiales de 6 bytes.

- Media aritmética (ms): 1964.77
- Desviación estándar (ms): 55.89

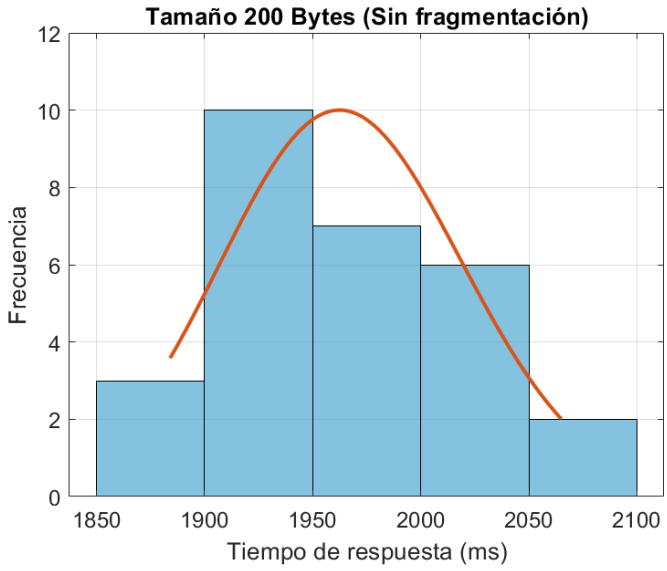

Figura 6.5: Histograma (azul) de los *pings* realizados para tamaño 200 bytes y densidad de probabilidad estimada (naranja)

El incremento de tiempo de respuesta medio es aproximadamente de 0.4 segundos mayor para un incremento de 110 bytes en el tamaño del paquete. Siguen siendo unos valores que se mueven en un rango razonable de *delay*, y que además mantienen una gran estabilidad obviando los paquetes perdidos por causas externas, siendo la desviación estándar bastante pequeña en comparación a la media aritmética (alrededor de un 3 %).

6.1.2. Con fragmentación

Habiendo probado un valor alto sin superar los 221 bytes que es el límite nuestra red LoRaWAN (para un *data rate* de 5 y un SF de 7), es momento de probar con valores superiores a dicho tamaño para probar la fragmentación de los paquetes IP y su envío y recepción.

El valor escogido para el nuevo tamaño de paquete es de 500 bytes, ampliamente superior al anterior pero todavía lejos del MTU (*Maximum Transmission Unit*) típico de una interfaz de red Ethernet. En este caso, ya sí es recomendable incrementar también el intervalo de *pings* para que no saturen las colas de envío y recepción. Además se añaderá en la configuración del ping para este caso un *timeout* mayor que el estándar, puesto que los resultados de tiempo de respuesta que se esperan serán mayores. Para esta configuración el comando utilizado es *ping -s 466 -c 50 -W 3000 -i 60 150.214.190.85*

Esta vez en el *log* del *forwarder* y del dispositivo LoRaWAN se espera que sea más de un fragmento el que se envíen y reciban para un sólo paquete IP. En la figura 6.6 se comprueba que para un paquete de 500 bytes el dispositivo LoRaWAN lo había fragmentado en tres partes de 221 bytes los dos primeros y 70 bytes el segundo. Esta fragmentación tiene sentido, porque tal y como se diseño este proceso (apartado 5.3.1) se espera un *payload* de 466 bytes y la suma de *payload* de cada fragmento según nuestro diseño debe ser de:

- *Payload* del primer fragmento = 221 bytes total −20 bytes cabeceras IP - 8 bytes cabeceras ICMP −6 bytes cabecera especial = 187 bytes.
- *Payload* del segundo fragmento = 221 bytes total −6 bytes cabecera especial = 215 bytes.
- *Payload* del último fragmento = 70 bytes total −6 bytes cabecera especial = 64 bytes.

Y la suma total es $187 + 215 + 64 = 466$ bytes de *payload.*

Otro aspecto interesante en el *log* del *forwarder* son los tiempos de recepción de cada fragmento, que suele ser aproximadamente de unos 4 segundos para cada fragmento de 221 bytes. Este valor debe ser una aproximación del tiempo en el aire del fragmento de 221 bytes con LoRaWAN.

Por otro lado, en el *Serial Monitor* de Arduino se encuentra menor detalle de información pero también se confirma el envío y recepción de tres fragmentos diferentes que se acaban formando en el paquete IP final que aparece entrecortado en la figura 6.7.

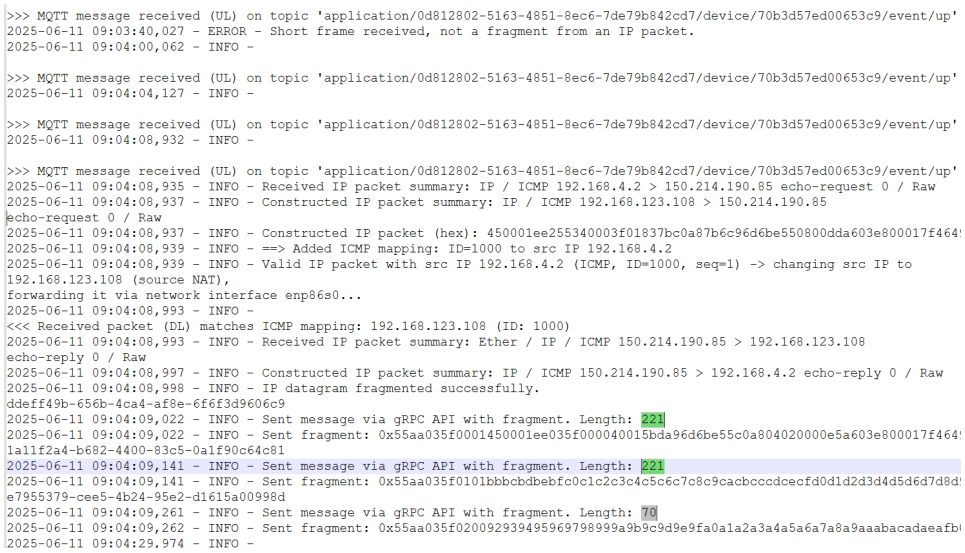

Figura 6.6: *Log* del *forwarder* aplicando fragmentación con tamaños de los fragmentos resaltados en color verde y gris

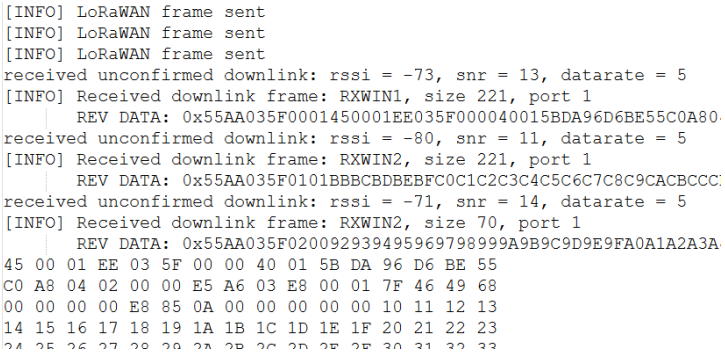

Figura 6.7: Serial monitor del dispositivo LoRaWAN aplicando fragmentación

Los resultados obtenidos del tiempo de respuesta para un tamaño de paquete IP de 500 bytes son los siguientes:

- Media aritmética (ms): 25860.35
- Desviación estándar (ms): 5718.86

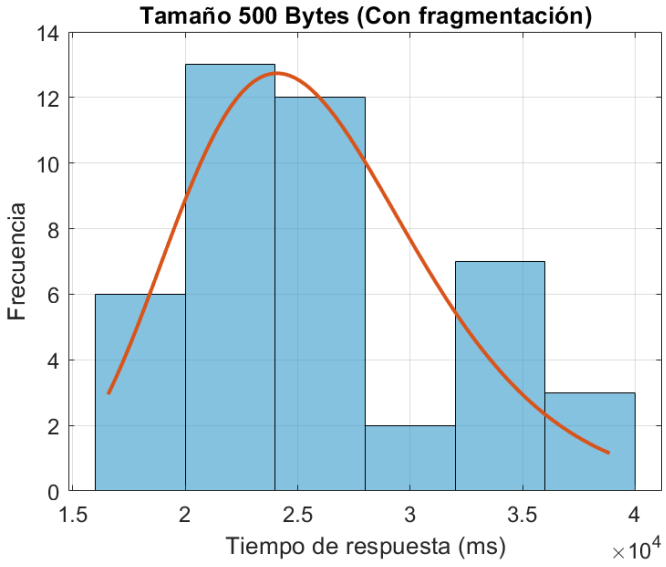

Figura 6.8: Histograma (azul) de los *pings* realizados para tamaño 500 bytes y densidad de probabilidad estimada (naranja)

En la figura 6.8 se concluye que el hecho de fragmentar el paquete IP aumenta considerablemente el tiempo de respuesta en la red, promediando unos 25 segundos por cada *ping*. Además la desviación estándar es cada vez mayor, siendo ahora un 22 % de la media aritmética y por lo tanto, los valores ahora están menos concentrados. Igualmente, es positivo que la fragmentación de paquetes IP sea funcional y que no se pierden muchos fragmentos en el aire (en este caso el porcentaje de paquetes perdidos es de un 20 %, no muy alejado del envío de paquetes sin fragmentación).

Por último, visto que la fragmentación se ejecuta de forma correcta, se va a llevar al límite el tamaño de paquete hasta lo que permite normalmente una MTU de una interfaz Ethernet, que es 1500 bytes. En este caso, se espera que la fragmentación sea de hasta 7 partes (siendo 6 de ellas de 221 bytes). También se realizan menos número de *pings* (30) debido a la larga espera entre cada uno de ellos. El comando a ejecutar es: *ping -s 1466 -c 30 -W 3000 -i 100 150.214.190.85.*

En la figura 6.9, se comprueba que el último fragmento es de 210 bytes, que si se repiten las cuentas anteriores:

- *Payload* del primer fragmento = 221 bytes total −20 bytes cabeceras IP - 8 bytes cabeceras ICMP −6 bytes cabecera especial = 187 bytes.
- *Payload* del segundo fragmento hasta el penúltimo fragmento = 221 bytes total −6 bytes cabecera especial = 215 bytes.
- *Payload* del último fragmento = 210 bytes total −6 bytes cabecera especial = 204 bytes.

Siendo la suma total de $187 + 215 * 5 + 204 = 1466$ bytes de *payload*, tal y como se ha indicado en la configuración del *ping*. Finalmente los resultados de tiempo de respuesta son los de la figura 6.10 y la desviación estándar:

- Media aritmética (ms): 67446.16
- Desviación estándar (ms): 3012.80

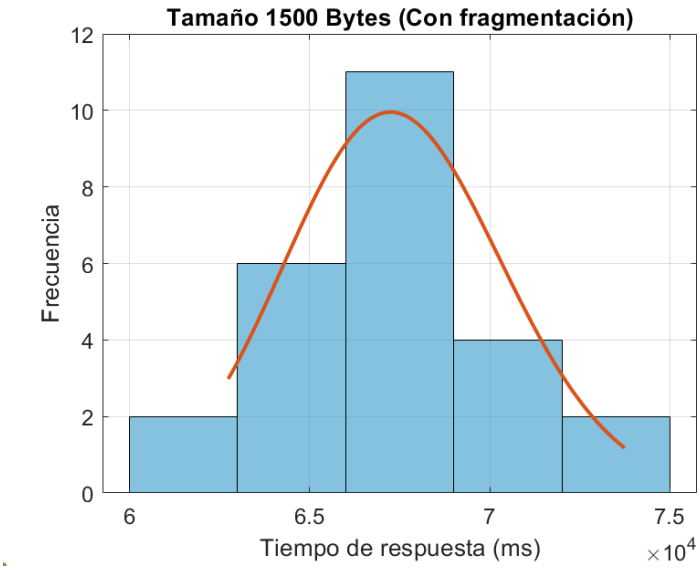

Figura 6.10: Histograma (azul) de los *pings* realizados para tamaño 1500 bytes y densidad de probabilidad estimada (naranja)

Para el tamaño máximo realizado en las pruebas se alcanzan tiempos de respuesta de hasta un minuto. Estos tiempos se siguen manteniendo bastante estables, incluso siendo en este caso la desviación estándar menor que

```
joining...
[INFO] LoRaWAN frame sent
joined
[INFO] Sending first LoRaWAN frame so we can receive in downlink...
[INFO] LoRaWAN frame sent
received unconfirmed downlink: rssi = -23, snr = 14, datarate = 5
[INFO] LoRaWAN frame sent
[INFO] LoRaWAN frame sent
[INFO] LoRaWAN frame sent
[INFO] LoRaWAN frame sent
[INFO] LoRaWAN frame sent
[INFO] LoRaWAN frame sent
[INFO] LoRaWAN frame sent
received unconfirmed downlink: rssi = -79, snr = 12, datarate = 5
[INFO] Received downlink frame: RXWIN1, size 221, port 1
       REV DATA: 0x55AA78A60001450005D678A600004001E2AA96D6BE55C0A8(
received unconfirmed downlink: rssi = -80, snr = 10, datarate = 5
[INFO] Received downlink frame: RXWIN2, size 221, port 1
       REV DATA: 0x55AA78A60101BBBCBDBEBFC0C1C2C3C4C5C6C7C8C9CACBCC(
received unconfirmed downlink: rssi = -71, snr = 13, datarate = 5
[INFO] Received downlink frame: RXWIN2, size 221, port 1
       REV DATA: 0x55AA78A602019293949596979899A9B9C9D9E9FA0A1A2A3A
received unconfirmed downlink: rssi = -80, snr = 9, datarate = 5
[INFO] Received downlink frame: RXWIN2, size 221, port 1
       REV DATA: 0x55AA78A60301696A6B6C6D6E6F707172737475767778797A7
received unconfirmed downlink: rssi = -78, snr = 11, datarate = 5
[INFO] Received downlink frame: RXWIN2, size 221, port 1
       REV DATA: 0x55AA78A6040140414243444546474849 4A4B4C4D4E4F5051
[INFO] LoRaWAN frame sent
received unconfirmed downlink: rssi = -76, snr = 12, datarate = 5
[INFO] Received downlink frame: RXWIN1, size 221, port 1
       REV DATA: 0x55AA78A605011718191A1B1C1D1E1F2021222324252627282
[INFO] LoRaWAN frame sent
received unconfirmed downlink: rssi = -88, snr = 3, datarate = 5
[INFO] Received downlink frame: RXWIN1, size 210, port 1
       REV DATA: 0x55AA78A60600EEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF(
45 00 05 D6 78 A6 00 00 40 01 E2 AA 96 D6 BE 55
C0 A8 04 02 00 00 0D 75 03 E8 00 01 A4 4C 49 68
00 00 00 00 E2 03 08 00 00 00 00 00 10 11 12 13
```

Figura 6.9: Serial monitor del dispositivo LoRaWAN aplicando y recibiendo fragmentación para 1500 bytes

en el caso anterior, pero realmente esto no es especialmente relevante. Lo más importante ha sido valorar y cuantificar el tiempo de respuesta del diseño de nuestra red y de haber podido mandar tráfico IP/IPv4 a través de LoRaWAN con diferentes tamaños de paquete. El cuadro 6.2 resume los resultados obtenidos de este apartado.

¿Requiere fragmentación?	**Tamaño del *Payload* (bytes)**	**Tiempo de respuesta promedio (ms)**
No	56	1565.15
No	166	1964.77
Sí	466	25860.35
Sí	1466	67446.16

Cuadro 6.2: Resumen de resultados de las pruebas ICMP

6.2. UDP y TCP

Para los protocolos UDP y TCP se va a realizar una prueba de funcionamiento con la creación de dos clientes (uno es el CPE y otro una máquina objetivo cualquiera) y un servidor UDP o TCP. El objetivo es lograr que los dos clientes sean capaz de recibir mensajes entre ellos a modo chat y también cualquier mensaje proveniente del servidor.

Para la creación del servidor y del cliente en la máquina destino se va a implementar un código en Python con la librería socket [33] para poder imprimir los eventos por pantalla de una forma mucho más cómoda. Este código asigna en la creación del cliente un nombre de usuario asociado a su dirección IP.

Por otro lado, se va a crear un cliente en el CPE (esta vez dispositivo móvil) con la aplicación *TCP UDP Server and Client* [34]. No requiere de root y se puede descargar de forma gratuita en Android a través de *Google Play Store*. En la figura 6.11 se puede ver la configuración utilizada, indicando la dirección del servidor IP (donde también se ubicará el otro cliente UDP) y el puerto que el servidor UDP abrirá para la conexión.

La interacción será la siguiente; una vez lanzado el servidor y el cliente (usuario *ethernet_user*) en la máquina anfitriona se recibirá un evento en el servidor de que dicho usuario se ha conectado. Posteriormente se unirá el dispositivo móvil con el usuario *lorawan_user* y desde ese momento podrán enviar y recibir mensajes UDP. En la figura 6.12 se ve cómo se disparan los eventos en el servidor y cómo se recibe el mensaje desde el dispositivo móvil (*lorawan_user*, con dirección IP cambiada mediante SNAT) y cómo responde el cliente (*ethernet_user*). Esto mismo también es visible en la figura 6.13.

Figura 6.11: Configuración cliente UDP desde la aplicación *TCP UDP Server and Client*

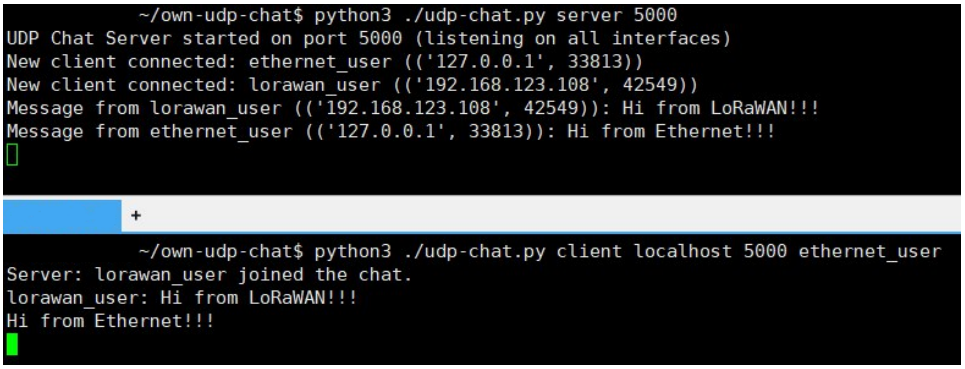

Figura 6.12: Lanzamiento del servidor y cliente en la máquina anfitriona (arriba servidor, abajo cliente).

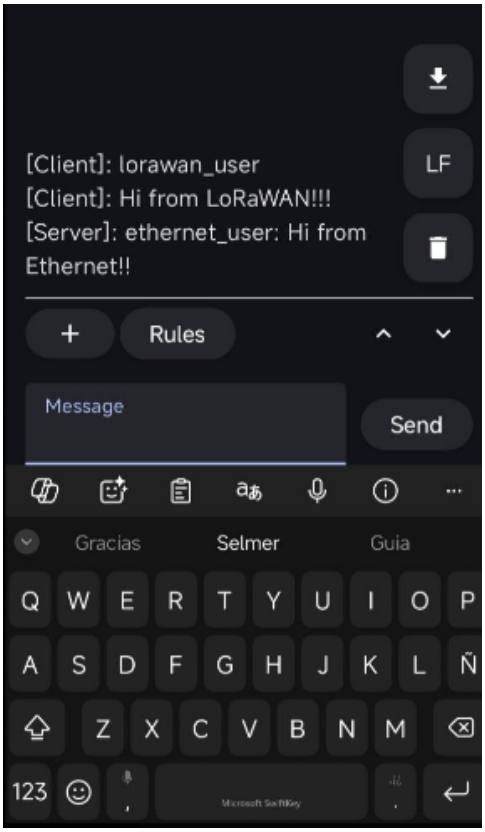

Figura 6.13: Intercambio de mensajes por chat UDP desde el móvil

Para justificar el correcto comportamiento de la red, se revisa el *log* del *forwarder* en la figura 6.14. Este *log* se corresponde al intercambio de mensajes previo y a la conexión del cliente desde el CPE al servidor UDP. Esto último ocurre en la primera sección del *log*, donde únicamente se conecta y notifica al servidor pero el servidor no vuelve a responder al CPE. En la segunda sección en cambio, el cliente sí envía un mensaje propio y se le reenvía una respuesta escrita por el *ethernet_user*. Este mensaje tiene una longitud de 68 bytes (incluyendo cabeceras especiales), por lo que su *delay*, que ya se ha repasado en la sección de ICMP, debe ser similar al 1-2 segundos.

Por cómo funciona UDP, en ningún momento es necesario algún tipo de respuesta por parte del servidor o del cliente porque en ningún instante hay alguna conexión establecida. Esto encaja bastante mejor con el comportamiento asíncrono de LoRaWAN y hace que UDP se sienta menos perjudicado en LoRaWAN que lo que se perjudican protocolos como ICMP o TCP.

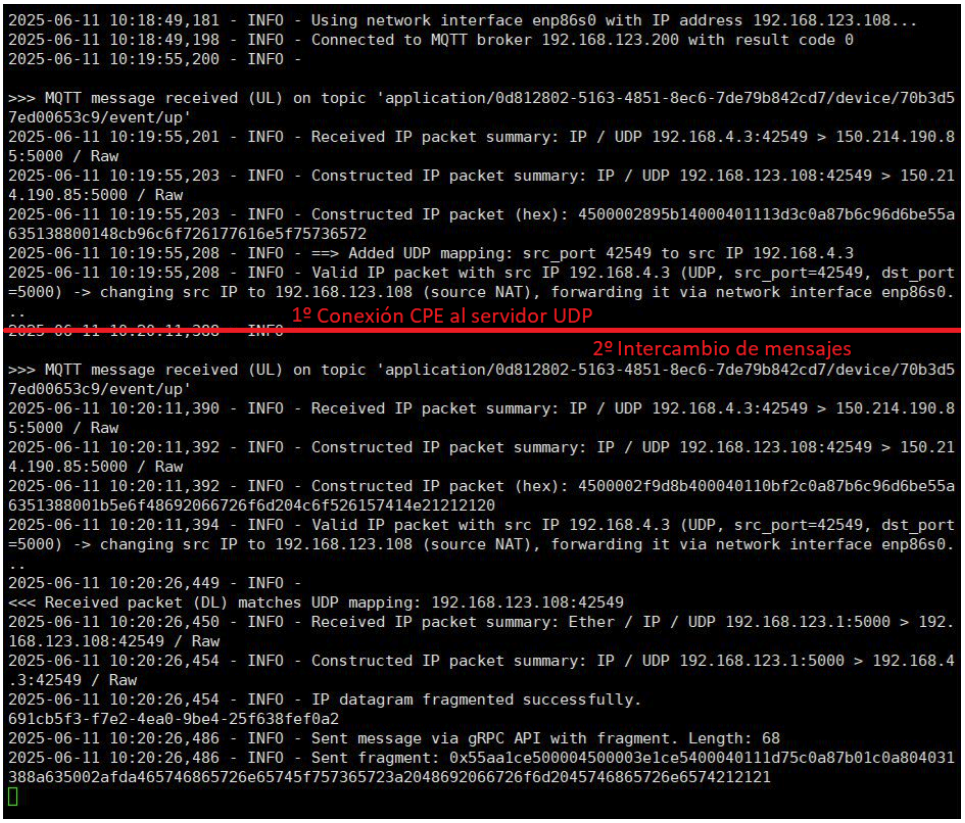

Figura 6.14: *Log* del *forwarder* en el intercambio de mensajes a través de un chat basado en UDP

Una última prueba que se ha planteado para UDP es la de simular un *iperf* para probar nuevamente la capacidad de la red pero esta vez utilizan-

```
Con nping usando 150 bytes (más cabeceras):
-------------------------------------------
13:40:24.518 -> [INFO] LoRaWAN frame sent
13:40:29.536 -> [INFO] LoRaWAN frame sent
13:40:33.608 -> [INFO] LoRaWAN frame sent
13:40:38.647 -> [INFO] LoRaWAN frame sent
13:40:44.679 -> [INFO] LoRaWAN frame sent
13:40:49.719 -> [INFO] LoRaWAN frame sent
13:40:53.801 -> [INFO] LoRaWAN frame sent
13:40:57.880 -> [INFO] LoRaWAN frame sent
13:41:03.915 -> [INFO] LoRaWAN frame sent
13:41:09.973 -> [INFO] LoRaWAN frame sent
13:41:14.983 -> [INFO] LoRaWAN frame sent
13:41:21.007 -> [INFO] LoRaWAN frame sent
13:41:26.055 -> [INFO] LoRaWAN frame sent
13:41:31.072 -> [INFO] LoRaWAN frame sent
13:41:35.167 -> [INFO] LoRaWAN frame sent
13:41:40.193 -> [INFO] LoRaWAN frame sent
13:41:45.197 -> [INFO] LoRaWAN frame sent
13:41:49.310 -> [INFO] LoRaWAN frame sent
13:41:54.302 -> [INFO] LoRaWAN frame sent
```

Figura 6.15: Envío de tramas LoRaWAN

do protocolo UDP. *Iperf* de por sí establece una conexión TCP previamente para intercambiar configuraciones y negociar parámetros. Por esto, directamente se ha optado por utilizar *nping* [35], que hace *pings* y genera paquetes a través del protocolo seleccionado (en este caso UDP). La idea es seleccionar un valor de tamaño de paquete sin fragmentación pero ciertamente alto y medir cada cuánto el dispositivo LoRaWAN es capaz de seguir recibiendo paquetes. El *nping* se realizará desde el cliente *ethernet_user* hacia el servidor, cada segundo.

En la figura 6.15 se obtiene un valor promedio de unos 4.76 segundos, que para 150 bytes de datos incluyendo cabeceras se obtiene un valor de:

Tasa aprox.(bps) = (150 bytes $*$8bits)/4.76segundos=**252bps**

A continuación se va a repetir el proceso pero esta vez creando un *socket* TCP. El proceso de creación del mismo es similar al anterior, únicamente hay que tener en cuenta nuevamente el puerto al que se va a conectar en el servidor para ajustarlo de la misma manera en el dispositivo LoRaWAN y también en la aplicación en el dispositivo móvil.

En la figura 6.16 se repite la misma interacción que para UDP, donde aparece en el servidor cómo se conectan ambos usuarios y también cómo acaba llegando el mensaje enviado a través del dispositivo móvil. Pero a diferencia de la prueba con UDP, y como se ve en la figura 6.17, *lorawan_user* sí se le envía de vuelta una respuesta en forma de ACK.

Figura 6.16: Lanzamiento del servidor y cliente en la máquina anfitriona (arriba servidor, abajo cliente). Establecimiento conexión TCP.

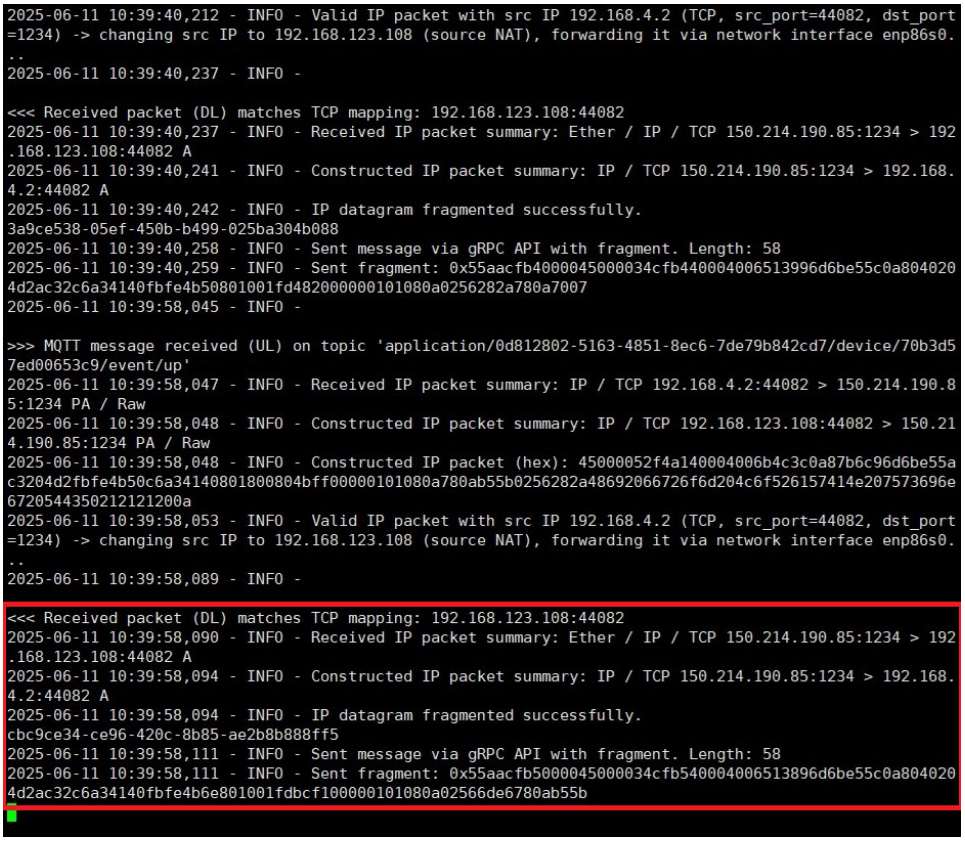

Figura 6.17: *Log* del *forwarder* en el envío de mensaje a través de un chat basado en TCP. En el cuadro rojo se marca el envío downlink para indicarle a *lorawan_user* que su mensaje ha llegado

Capítulo 7

Conclusiones y Trabajos Futuros

En este último capítulo se revisan las conclusiones obtenidas de forma general en los distintos capítulos y en el proyecto en su totalidad. También se detallarán ciertas líneas de trabajo que podrían ser interesantes para mejorar el rendimiento y la funcionalidad de este proyecto.

7.1. Conclusiones

En este proyecto se ha diseñado e implementado una arquitectura de red, capaz de enviar cierto tráfico IP/IPv4 a través de LoRaWAN. Para conseguir esto se han tenido que diseñar e implementar funcionalidades en ciertos componentes de la red para que pudieran aceptar tráfico de protocolos como ICMP, UDP o TCP. Estas funcionalidades han debido manipular los paquetes de los distintos protocolos para que sean aptos de poder enviarse por un protocolo de red tan diferente como LoRaWAN.

Los resultados de las implementaciones han sido mayormente exitosos, aceptando paquetes ICMP de diferentes tamaños, incluso superiores a lo que LoRaWAN puede aceptar. A pesar de que el retardo es alto, esto ya era una característica propia de LoRaWAN que simplemente se ha visto aún más afectada, pero a costa de perjudicar esta característica ya de por sí negativa, se ha conseguido establecer un tipo de comunicación que hasta ahora no se había implementado en este tipo de redes. Esto también se ve reflejado en las pruebas para UDP y TCP, donde el resultado obtenido especialmente para UDP es bastante satisfactorio, siendo capaz de establecer un sistema de mensajería para el usuario.

Gracias a haber realizado estas pruebas, se puede concluir que es posi-

ble enviar ciertas cantidades de información con dispositivos convencionales como un móvil que no están preparados para enviar tráfico por LoRaWAN, y comunicarse con otras máquinas en espacios sin ningún tipo de cobertura móvil. Ser capaces de poder estar comunicados en este tipo de situaciones puede ser crucial y de gran ayuda en casos de emergencia donde las comunicaciones convencionales no están disponibles.

7.2. Trabajos futuros

Por último, se listan una serie de posibles mejoras, implementaciones o ideas que se pueden añadir en un futuro para poder hacer un proyecto más sólido:

- Retransmisiones: Añadir la capacidad de retransmisión de paquetes en caso de pérdida (especialmente de fragmentos) es una idea que no sólo no debe ser especialmente complejo de implementar, si no que puede ser interesante para hacer una comunicación más robusta y evitar un gran número de pérdidas en paquetes que sean muy grandes y requieran de una mayor fragmentación.
- Realizar un mayor número de medidas: Habiendo implementado lo anterior, realizar un mayor número de medidas puede ayudar a conocer mejor las capacidades de la red. El único requisito es disponer de mucho tiempo para dejar funcionando la red y poder evaluarla mejor.
- Cambiar el escenario de pruebas: Como se ha comentado, el escenario de este proyecto ha sido en un entorno de laboratorio o en su defecto doméstico. Probar las capacidades de la red en un escenario más apropiado al que se pretende utilizar como en una montaña o zona sin cobertura es una tarea pendiente para el futuro.
- Implementación de compresión de cabeceras: Aunque el reto principal es el de la fragmentación de paquetes, si se consigue comprimir las cabeceras IP mediante algún tipo de compresor como ROHC para IPv4, se puede reducir el tamaño de los paquetes. En ciertos casos esto puede evitar fragmentación de paquetes y reducir considerablemente el tiempo de respuesta.
- Gestión de colas: Implementar o mejorar un gestor de colas en el dispositivo LoRaWAN puede mejorar la carga del dispositivo, ampliando la vida útil del mismo y haciendo la red algo más flexible.

Apéndice A

Anexo I. Manual de Instalación

La instalación del proyecto se ha realizado sobre una máquina con sistema operativo Windows 10 que dispone de una máquina virtual en Ubuntu 22.04 con las dependencias actualizadas. Sobre este último sistema operativo es donde se instalará Chirpstack y se ubicará el componente *forwarder*. Es importante tener en cuenta que si se elige esta opción se debe configurar una redirección de puertos para la máquina virtual, para que ambas máquinas tengan en común los puertos 1700 para la gateway, 8080 para la GUI de Chirpstack y 1883 para MQTT.

Lo primero es instalar Docker y Docker Compose [36] (quizá ya se descargó actualizado dependencias de Ubuntu 22.04). Nótese también que la mayoría de complementos de Chirpstack como PostgreSQL, Redis y Mosquitto ya vienen incluidos en la instalación de Chirpstack mediante *docker*, lo que hace la instalación de la plataforma muy sencilla y cómoda. Se puede clonar el repositorio oficial de Chirpstack:

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

Una vez clonado para ponerlo en funcionamiento únicamente hay que lanzar el contenedor (última versión de Docker Compose va sin guión):

```
docker compose up -d
```

Con esto ya se tendría lanzada la plataforma Chirpstack que se puede consultar su GUI en cualquier navegador para localhost:8080 y configurar la puerta de enlace y dispositivos LoRaWAN necesarios tal y como se ha explicado en el capítulo de diseño e implementación.

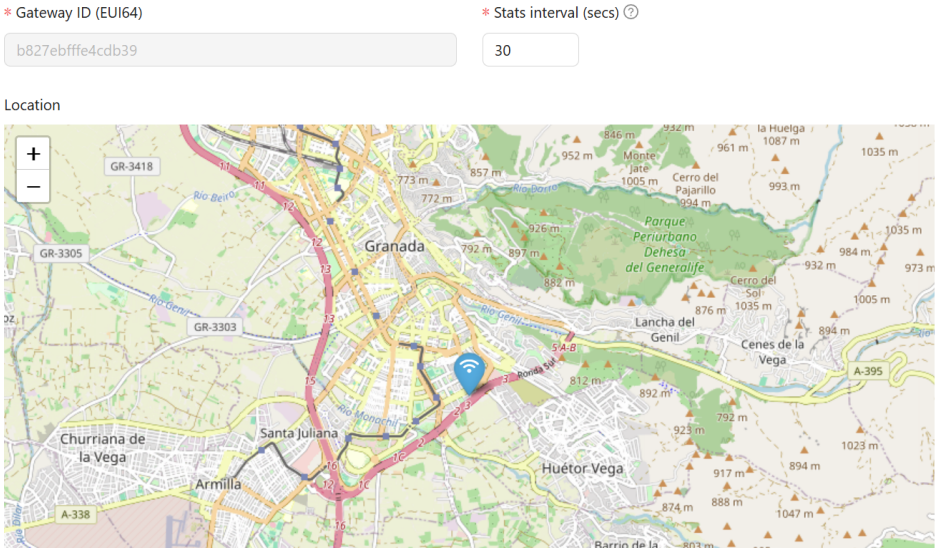

Figura A.1: Gateway configurado y disponible en plataforma Chirpstack

Una vez lanzado se ha de crear un perfil (con las indicaciones de la sección 5.2) y añadir el gateway indicando su ID. Se conocerá que está online cuando lo indique nuestra plataforma y seamos capaces de ver la ubicación del mismo (figura A.1. El gateway debe estar conectado de alguna manera a la máquina anfitriona del servidor Chirpstack, se puede hacer mediante cable Ethernet y configurando las direcciones IPv4 para que se puedan comunicar. Para lanzar el gateway se accede a su terminal por SSH y se ejecuta:

```
./start_gateway.sh
```

Para la creación de nuestra aplicación se deben incluir los dispositivos LoRaWAN relacionados, esto significa indicar su Device EUI, el perfil que utilizan anteriormente creado y especificar las claves OTAA. Nuestro dispositivo LoRaWAN funciona como un plug-and-play porque ya tiene el código integrado y se ejecutará nada más lo conectemos a cualquier USB. Si se ha realizado correctamente la configuración e instalación aparecerá en el apartado *Events* como *uplink* un mensaje de *Join Request* por parte del dispositivo LoRaWAN e información relativa a qué aplicación y perfil pertenece dentro de nuestra plataforma.

También en el apartado LoRaWAN Frames se puede encontrar información más precisa sobre el mensaje recibido por el servidor en el *uplink*, así como sus características de transmisión y recepción, payload y características LoRa (figura A.2).

Para el forwarder únicamente hay que ubicar la carpeta en la misma

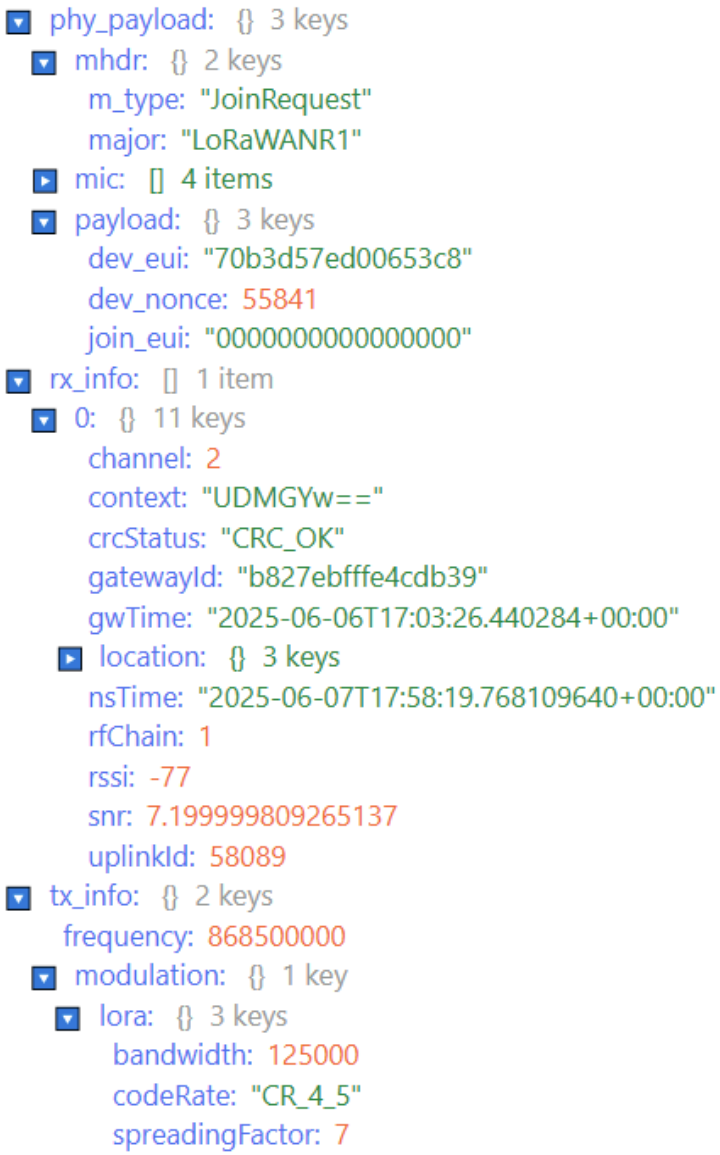

Figura A.2: Trama LoraWAN (Join Request) mandada por el dispositivo LoRaWAN a la plataforma Chirpstack

máquina anfitriona de la plataforma Chirpstack (aunque no es estrictamente necesario) donde se encontrarán dos versiones *forwarder.sh o forwarder_woTLS.sh*. La diferencia es que una utiliza TLS para MQTT y mosquitto y el segundo lo obvia. Los pasos de esta instalación optan por la segunda opción por comodidad pero es totalmente válido y posible implementar TLS.

Es importante editar los archivos de configuración que utiliza el *forwarder.sh* que son necesarios para establecer conexión con el servidor Chirpstack y suscribirse a los topic necesarios, estos archivos son APISERVER.txt para la dirección IP de la plataforma Chirpstack (si es en la misma máquina será localhost), APITOKEN.txt que es el token generado por Chirpstack para el funcionamiento de gRPC API, TOPIC_RX y TOPIC_TX para indicar los tópicos a los que se tiene que suscribir en uplink y downlink respectivamente, MQTTPORT.txt que variará entre 1883 sin TLS y 8883 para TLS, MQTTSERVER.txt que es la dirección del MQTT broker (similar a la de Chirpstack) y finalmente MQTTPASS y MQTTUSER para indicar usuario y contraseña creados en caso de TLS, además de incluir el certificado de seguridad en la misma carpeta que el código *forwarder*.

Las librerías Python necesarias para el correcto funcionamiento del *forwarder* son paho-mqtt, grpcio, chirpstack-api y scapy:

```
pip3 install paho-mqtt
pip3 install grpcio
pip3 install chirpstack-api
pip3 install scapy
```

Apéndice B

Anexo II. Manual de Usuario

Una vez instalado y configurado todo (A), se lanza y se despliega la arquitectura de red empezando por la plataforma Chirpstack en caso de que se hubiese cerrado anteriormente.

```
docker compose up -d
```

Cuando esté lanzado, se accede a la terminal del gateway mediante SSH para lanzar el script interno para su funcionamiento básico.

```
./stop_gatewayh.sh #en caso de que ya estuviese funcionando
./start_gateway.sh
```

Esperando unos segundos y comprobando que el gateway aparece online en la GUI de Chirpstack, se ejecuta el *forwarder* en segundo plano.

```
./forwarder_woTLS.sh #o ./forwarder.sh en caso de usar TLS
```

Una vez ejecutado el *forwarder* se puede conectar el dispositivo LoRaWAN a cualquier USB para que empiece a funcionar y desplegar el punto de acceso wifi. Ya en la plataforma LoRaWAN se verá en los eventos un *Join Request*, mientras que en el *forwarder* aparecerá que ha llegado un mensaje que no es de IP al tópico al que está suscrito (refiriéndose a este mismo *Join Request*).

Por otro lado, se conecta el disopsitivo LoRaWAN de nuevo que automáticamente creará el punto de acceso Wifi. Cuando el punto de acceso esté creado nos unimos con el móvil y se comprueba que le asigna una dirección IP mediante DHCP. Desde Arduino IDE se puede hacer debug de todo el proceso y comprobar qué dispositivos se han unido.

Figura B.1: Serial monitor del dispositivo LoRaWAN. Envía un trama LoRaWAN para Join Request y crea el punto de acceso wifi, el dispositivo móvil se une y le asigna la dirección 192.168.4.2

Es recomendable haber activado previamente algún tipo de cortafuegos en el dispositivo móvil para evitar mayor tráfico del necesario y así poder mantener controlado el entorno de prueba. En este proyecto se utiliza NetGuard que es una aplicación gratuita que se pueden encontrar en Google PlayStore y realiza de cortafuegos sin necesidad de root (figura B.2).

Con esto ya estaría todo dispuesto para iniciar la comunicación. Un ejemplo sencillo ICMP se puede ejecutar desde el móvil con la aplicación gratuita PingTools sin necesidad de root. La dirección de destino para hacer el la prueba de tráfico ICMP puede ser cualquier máquina que esté conectada a la máquina anfitriona del forwarder, en este caso, se ha escogido el propio gateway como ejemplo de máquina destino por comodidad. En la figura B.3

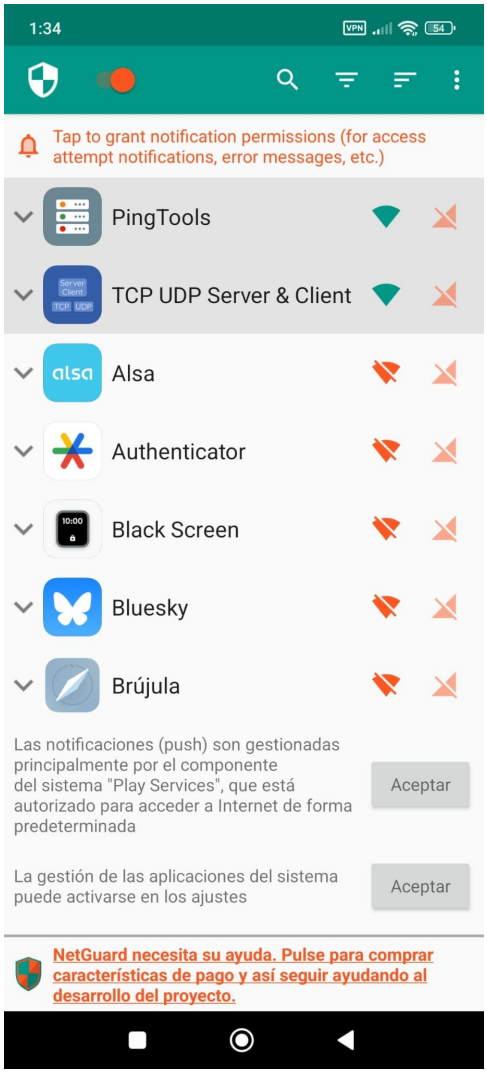

Figura B.2: Uso de cortafuegos mediante NetGuard. Se permite el envío de tráfico únicamente para las aplicaciones que se utilizarán en las pruebas.

Figura B.3: *Ping* realizado con un dispositivo móvil a la dirección IP del *gateway* como prueba de funcionamiento

Bibliografía

[1] *State of IoT 2024: Number of connected IoT devices growing 13 to 18.8 billion globally — iot-analytics.com*, https://iot-analytics.com/number-connected-iot-devices/, [Accessed 05-06-2025].

[2] A. CyberSecurity, *IP2Lora*, https://github.com/airbus-cyber/IP2LoRa, 2020.

[3] R. T. Limited, *WisDuino — store.rakwireless.com*, https://store.rakwireless.com/collections/wisduino?srsltid=AfmBOorNkk_oZIb4iqh4XZV-mYQEgEz4jj2tb7OUDlRqbsua5YLfgtvf, [Accessed 20-05-2025].

[4] *LoStik - USB LoRa device — ronoth.com*, https://ronoth.com/products/lostik, [Accessed 20-05-2025].

[5] J. Sakamoto, D. Nobayashi, K. Tsukamoto, T. Ikenaga, G. Sato y K. Takizawa, "Improvement of TCP Performance based on Characteristics of Private LoRa Interface," en *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2023, págs. 998-999. DOI: 10.1109/COMPSAC57700.2023.00145.

[6] A. Yegin, *IPv6 Officially Enters the LoRaWAN Ecosystem — blog.semtech.com*, https://blog.semtech.com/ipv6-officially-enters-the-lorawan-ecosystem, [Accessed 05-06-2025].

[7] *Wireless Stick(V3), ESP32S3 + SX1262 LoRa Node, Meshtastic and LoRaWAN Compatible — heltec.org*, https://heltec.org/project/wireless-stick-v3/, [Accessed 19-05-2025].

[8] *Lite Gateway - Demonstration Platform for LoRa Technology — shop.imst.de*, https://shop.imst.de/wireless-solutions/lora-products/36/lite-gateway-demonstration-platform-for-lora-technology, [Accessed 19-05-2025].

[9] *Adaptador de red USB 3.0 a Gigabit Ethernet — tp-link.com*, https://www.tp-link.com/es/home-networking/adapter/ue300/, [Accessed 20-05-2025].

[10] *Arduino - Home — arduino.cc*, https://www.arduino.cc/, [Accessed 20-05-2025].

[11] *Welcome to Python.org — python.org*, https://www.python.org/, [Accessed 20-05-2025].

[12] *Introduction - ChirpStack Open-Source LoRaWAN Network Server documentation — chirpstack.io*, https://www.chirpstack.io/docs/, [Accessed 13-05-2023], 2016.

[13] *Oracle VirtualBox — virtualbox.org*, https://www.virtualbox.org/, [Accessed 20-05-2025].

[14] B. S. s.r.o, *GanttProject: free project management tool for Windows, macOS and Linux — ganttproject.biz*, https://www.ganttproject.biz/, [Accessed 20-05-2025].

[15] *Download PuTTY - a free SSH and telnet client for Windows — putty.org*, https://www.putty.org/, [Accessed 20-05-2025].

[16] *Overleaf, Online LaTeX Editor — overleaf.com*, https://www.overleaf.com/, [Accessed 20-05-2025].

[17] *US9647718B2 - Wireless communication method - Google Patents — patents.google.com*, https://patents.google.com/patent/US9647718B2/en, [Accessed 05-06-2025].

[18] J. Liando, A. Gamage, A. Tengourtius y M. Li, "Known and Unknown Facts of LoRa: Experiences from a Large-scale Measurement Study," *ACM Transactions on Sensor Networks*, vol. 15, págs. 1-35, feb. de 2019. DOI: 10.1145/3293534.

[19] *ITU-T Recommendation database — itu.int*, https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=14818&lang=es, [Accessed 05-06-2025].

[20] *RF Regions — ThingPark Enterprise All-in-one — docs.thingpark.com*, https://docs.thingpark.com/thingpark-enterprise-all-in-one/configuration-usage/rf-regions-configuration/rf-regions, [Accessed 16-06-2025].

[21] *Regional Limitations of RF Use in LoRaWAN — thethingsnetwork.org*, https://www.thethingsnetwork.org/docs/lorawan/regional-limitations-of-rf-use/, [Accessed 16-06-2025].

[22] A. Tesfay, "Dealing with interfering users in LoRa Networks," Tesis doct., oct. de 2021.

[23] H. Noura, T. Hatoum, O. Salman, J.-P. Yaacoub y A. Chehab, "LoRaWAN security survey: Issues, threats and possible mitigation techniques," *Internet of Things*, vol. 12, pág. 100 303, 2020, ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100303. dirección: https://www.sciencedirect.com/science/article/pii/S2542660520301359.

[24] *MQTT man page — mosquitto.org*, https://mosquitto.org/man/mqtt-7.html, [Accessed 13-05-2025].

[25] *Documentation — mosquitto.org*, https://mosquitto.org/documentation/, [Accessed 13-05-2025].

[26] *Socket API — zeromq.org*, https://zeromq.org/socket-api/, [Accessed 13-11-2024].

[27] *Welcome to Scapy documentation! Scapy 2.6.1 documentation — scapy.readthedocs.io*, https://scapy.readthedocs.io/en/latest/, [Accessed 13-05-2025].

[28] *Class C Downlink issue in v4 — forum.chirpstack.io*, https://forum.chirpstack.io/t/class-c-downlink-issue-in-v4/18963, [Accessed 07-06-2025].

[29] I. Craggs, *Eclipse Paho — The Eclipse Foundation — eclipse.dev*, https://eclipse.dev/paho/, [Accessed 15-05-2025].

[30] *Protocol documentation - ChirpStack open-source LoRaWAN; Network Server documentation — chirpstack.io*, https://www.chirpstack.io/docs/chirpstack/api/api.html, [Accessed 16-06-2025].

[31] *Wi-Fi - ESP32 - 2014; ESP-IDF Programming Guide v5.4.1 documentation — docs.espressif.com*, https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_wifi.html, [Accessed 26-05-2025].

[32] *Hex Packet Decoder — hpd.gasmi.net*, https://hpd.gasmi.net/, [Accessed 13-06-2025].

[33] *socket — Low-level networking interface — docs.python.org*, https://docs.python.org/3/library/socket.html, [Accessed 13-06-2025].

[34] *TCP UDP Server & Client - Apps en Google Play — play.google.com*, https://play.google.com/store/apps/details?id=tcpudpserverclient.steffenrvs.tcpudpserverclient&hl=es_419, [Accessed 13-06-2025].

[35] *nping(1) - Linux manual page — man7.org*, https://man7.org/linux/man-pages/man1/nping.1.html, [Accessed 13-06-2025].

[36] *Linux — docs.docker.com*, https://docs.docker.com/desktop/setup/install/linux/, [Accessed 06-06-2025].

[37] J. Liando, A. Gamage, A. Tengourtius y M. Li, "Known and Unknown Facts of LoRa: Experiences from a Large-scale Measurement Study," *ACM Transactions on Sensor Networks*, vol. 15, págs. 1-35, feb. de 2019. DOI: 10.1145/3293534.

[38] A. Tesfay, *Dealing with interfering users in LoRa Networks*, oct. de 2021.

[39] H. Noura, T. Hatoum, O. Salman, J.-P. Yaacoub y A. Chehab, "LoRaWAN security survey: Issues, threats and possible mitigation techniques," *Internet of Things*, vol. 12, pág. 100 303, 2020, ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100303. dirección: https://www.sciencedirect.com/science/article/pii/S2542660520301359.

[40] R. Kufakunesu, G. P. Hancke y A. M. Abu-Mahfouz, "A Survey on Adaptive Data Rate Optimization in LoRaWAN: Recent Solutions and Major Challenges," *Sensors*, vol. 20, n.º 18, 2020, ISSN: 1424-8220. DOI: 10.3390/s20185044. dirección: https://www.mdpi.com/1424-8220/20/18/5044.

[41] S. Yalçin, "An artificial intelligence-based spectrum sensing methodology for LoRa and cognitive radio networks," *International Journal of Communication Systems*, vol. 36, ene. de 2023. DOI: 10.1002/dac.5433.