ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

Plataforma de Pruebas de Conformidad LoRaWAN

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Pablo Daniel Modernell Echenique

en cumplimiento parcial de los requerimientos para la obtención del título de Magister en Ingeniería Eléctrica.

Directores de Tesis

Dr. Leonardo Steinfeld ..... Universidad de la República
Dr. Xavier Vilajosana ..... Universitat Oberta de Catalunya

Tribunal

Dr. Germán Capdehourat ..... Universidad de la República
Dr. Diego Dujovne ..... Universidad Diego Portales, Chile
Dr. Pere Tuset ..... Universitat Oberta de Catalunya, España

Director Académico

Dr. Leonardo Steinfeld ..... Universidad de la República

Montevideo
jueves 15 octubre, 2020

Plataforma de Pruebas de Conformidad LoRaWAN, Pablo Daniel Modernell Echenique.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 164 páginas.
Compilada el jueves 15 octubre, 2020.
http://iie.fing.edu.uy/

Agradecimientos

Quiero comenzar agradeciendo a mis directores de tesis, Leonardo Steinfeld y Xavier Vilajosana, trabajar con ellos ha sido una experiencia magnífica por su calidad humana y enorme valor profesional. Leonardo me ha acompañado desde el inicio de la Maestría con un aporte muy valioso en la elección de los cursos y la definición de la tesis, este documento ha mejorado muchísimo gracias a sus aportes y correcciones. Agradezco la confianza de Xavier al invitarme a participar del proyecto F-LoRa que da inicio a esta tesis, así como por su apoyo cada vez que necesité ayuda para avanzar en el mismo. Su dedicación y preocupación porque quienes trabajan con él tengan las mejores condiciones posibles están a la par de su capacidad profesional.

La integración de la plataforma en el entorno de pruebas F-Interop contó con la colaboración de Remy Leone y Tengfei Chang desde el instituto Inria en París, agradezco sus aportes así como la amabilidad con la que me recibieron en París tanto ellos cómo Thomas Watteyne y el resto de su equipo. También quiero mencionar y agradecer a Federico Sismondi, quien desde su rol de Test Specialist para F-Interop me facilitó mucho el trabajo.

Parte de este trabajo de tesis fue realizado en el edificio del IN3 de la UOC, y no quiero dejar de mencionar a Borja Martinez y Cristina Cano. Les agradezco por sus comentarios y sugerencias cuando les presenté el trabajo al finalizar el proyecto F-LoRa así como por crear un ambiente de trabajo inmejorable en el IN3.

Agradezco a la Agencia Nacional de Investigación e Innovación por la asignación de mi beca de maestría, así como a todas las personas que apoyan y luchan por defender a la Universidad de la República, los sistemas de financiamiento a la investigación y la educación pública en el Uruguay.

Esta tesis pone fin a mi Maestría en Ingeniería Eléctrica, un tipo de emprendimiento que requiere de esfuerzo y dedicación personales que son irremplazables, pero sólo son realizables si se dan las condiciones y el entorno que hacen posible que uno pueda dedicarse a ello. Siento un enorme agradecimiento hacia todas las personas que ayudaron a que esto sea posible. Agradezco profundamente a Gabriela por su apoyo, por estar ahí y ayudar a crear las condiciones para que mi maestría sea posible. Agradezco a mi familia por su apoyo, a Sergio, a Trinidad, a mis tíos Antonio y Martha, a mis abuelos, a mi madre, y a toda la gente que ayudó a crear las condiciones para que yo hoy pueda terminar esta maestría. Hicieron que todo fuera más fácil, y en algunos casos sin su apoyo esto hubiera sido literalmente imposible.

Esta página ha sido intencionalmente dejada en blanco.

A Inari y Bamboo.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El protocolo de comunicación LoRaWAN definido por la LoRa Alliance se destaca entre las Low Power Wide Area Networks, redes de bajo consumo y largo alcance, ya que ha facilitado el desarrollo de aplicaciones para ciudades inteligentes, manejo de residuos o agricultura de precisión.

El crecimiento de la adopción de dispositivos inalámbricos enfatiza la importancia de la estandarización asegurando la compatibilidad entre fabricantes. Para que una tecnología pueda ser ampliamente adoptada deben alinearse las visiones de los actores involucrados, definiendo procesos que verifiquen que una implementación se desarrolló en conformidad con el estándar. Es la LoRa Alliance quien define un proceso y un conjunto de pruebas a las que debe ser sometido un dispositivo para poder afirmar que es compatible con LoRaWAN.

Una forma de acortar los tiempos de lanzamiento al mercado de un producto, y evitar que los desarrolladores tengan que implementar sus propias pruebas, es mediante herramientas de pre-certificación que permitan verificar en etapas tempranas del desarrollo que una implementación cumple con el estándar.

Este trabajo estudia las principales características de LoRaWAN y desarrolla una plataforma de pruebas de conformidad. Una arquitectura basada en servicios comunicándose a través de un broker central de mensajería permite que la plataforma pueda ejecutarse de forma local en un PC del usuario y también integrarse en otros entornos de pruebas. La plataforma desarrollada fue integrada en el proyecto europeo F-Interop como la herramienta F-LoRa.

Con un diseño modular y extensible, se enfoca el desarrollo en la región europea, permitiendo añadir tests que prueben otras funcionalidades de LoRaWAN o que extiendan el alcance soportando características de otras regiones.

Se obtuvieron resultados satisfactorios al evaluar la plataforma con un dispositivo LoRaWAN certificado, donde se pudo comprobar que el manejo de sesiones, la encriptación y el envío de comandos fueron correctamente implementados en la plataforma. Adicionalmente, se realizaron modificaciones a una implementación certificada para inyectarle errores y evaluar cómo estos son detectados.

Se espera que este desarrollo aporte valor principalmente en entornos educativos y para situaciones donde contar con el control total del entorno de pruebas resulte interesante.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

El presente trabajo de tesis aborda la temática de las redes de comunicación de largo alcance y bajo consumo, enfocándose en el desarrollo de una plataforma de pruebas de conformidad para el estándar LoRaWAN. En este capítulo se realiza una breve introducción al contexto de aplicación de este tipo de tecnologías, fundamentando la importancia de la realización de tests de conformidad en etapas tempranas de la implementación de los módulos de comunicación de un dispositivo.

En las siguientes secciones se especifican los objetivos, alcance y limitaciones de la plataforma desarrollada y se realiza una introducción al proyecto F-Interop, modelo usado como referencia y entorno de pruebas donde se ha integrado la plataforma desarrollada cómo un módulo de tests de conformidad para LoRaWAN.

1.1. Contexto de Aplicación de LoRaWAN

El ámbito de aplicación de LoRaWAN se enmarca dentro de las denominadas Low Power Wide Area Networks (LPWAN), redes de bajo consumo y largo alcance que se proyectan para dar soporte a una gran parte de los dispositivos conectados en la Internet of Things (IoT) [15]. Estas tecnologías se vuelven relevantes en escenarios donde por ejemplo otras cómo Bluetooth, BLE (Bluetooth Low Energy), Wi-Fi o ZigBee no son adecuadas por su corto alcance y algunas tecnologías celulares resultan costosas o requieren un consumo excesivo de energía [18] [27].

En lo que refiere a las tecnologías celulares, la respuesta para cubrir las necesidades de bajo consumo de las LPWAN ha sido la definición del estándar Narrow Band Internet of Things (NB-IoT) cómo extensión de LTE (Long Term Evolution). NB-IoT logra alcanzar los niveles de LoRaWAN en sus características de bajo consumo y además permite ofrecer mayor confiabilidad gracias a la utilización de bandas de frecuencia dedicadas, que al no tener que compartirse con usuarios externos permite una mejor planificación de los recursos. Por otra parte, usar esta tecnología implica la dependencia respecto a un operador que es el dueño de la red y cobra en base a los datos transmitidos [20].

En términos generales estas son las principales características de las LPWAN [18]:

- Largo alcance, en el entorno de los 10 km entre emisor y receptor en entornos

abiertos.

- Baja tasa de transferencia de datos, típicamente con mensajes de unas pocas decenas de bytes enviados algunas veces al día.

- Bajo consumo, con dispositivos que pueden funcionar sin cambiar las baterías por períodos de entre 5 y 10 años.

El largo alcance y penetración en entornos urbanos de este tipo de tecnologías de comunicación, logrado generalmente empleando frecuencias de las bandas ISM (Industrial, Scientific and Medical) en el rango de sub-GHz, permiten la adopción de topologías de tipo estrella. Los dispositivos envían directamente los datos a un concentrador o gateway central simplificando la gestión de la red y bajando el costo de infraestructura [27].

Ejemplos donde las LPWAN muestran un gran potencial son la conexión de dispositivos en ciudades inteligentes, detección y gestión de parking de vehículos, manejo de residuos o edificios inteligentes [8] [1]. Un sector con una creciente adopción de estas tecnologías de comunicación es el de la logística, con un importante número de aplicaciones de rastreo y monitorización en la cadena de suministros. Información relevante sobre el estado y ubicación de bienes (e.g. temperatura y coordenadas) puede ser transmitida utilizando una tasa de transmisión de datos baja y no existen restricciones importantes de latencia [7].

Otra aplicación donde se espera que las LPWAN jueguen un rol importante es en la agricultura de precisión. El aumento de la demanda en la producción de alimentos y la necesidad de reducir el impacto a los ecosistemas requieren de la modernización y utilización eficiente y sostenible de los recursos. Una modernización hacia una agricultura inteligente implica la instalación de sensores a gran escala que permitan monitorear el estado de los cultivos (e.g. temperatura, humedad, niveles de polinización, etc.), teniendo en cuenta las grandes estimaciones de crecimiento en un mercado donde menos de un 2 % de la tierra cultivada cuenta con sensores instalados [9] [16].

LoRaWAN y en general las LPWAN no resultan adecuadas en situaciones que requieran de tasas de transferencia de datos elevadas o respuestas en tiempo real. Ejemplos de este tipo pueden encontrarse en aplicaciones de video-vigilancia, o aplicaciones de automatización industrial donde existan restricciones en los tiempos de respuesta aceptables [1].

LoRaWAN, que define una arquitectura de red y capa MAC operando sobre enlaces con modulación LoRa [29], tal como se explica en el Capítulo 2, es una de las tecnologías de LPWAN más adoptadas [31] y puede diferenciarse desde el punto de vista técnico y desde la perspectiva de su modelo de negocios. Una de las principales diferencias se encuentra en los menores costos de infraestructura respecto a NB-IoT (utilización de bandas no licenciadas) y facilidad para realizar despliegues independientes [6].

1.2. Estandarización y Pruebas de Conformidad

En los últimos años se observa un crecimiento acelerado de las conexiones y tráfico de datos entre máquinas (M2M, Machine-to-Machine) con acceso a Internet e integración con aplicaciones de procesamiento de datos [14]. La necesidad de interconexión de diferentes sistemas pone de manifiesto la importancia de la estandarización y compatibilidad entre estos, ya que uno de los principales retos para reducir los tiempos de desarrollo y facilitar economías de escala está en la fragmentación de tecnologías. Esta fragmentación hace referencia a la gran cantidad de tecnologías de comunicación coexistiendo [33] [32].

Para que una tecnología pueda ser ampliamente adoptada, antes del lanzamiento al mercado se requiere que el producto desarrollado esté probado frente a un estándar. Los procesos previos de estandarización, donde se logra alinear los intereses y visiones de los actores involucrados, y validación de la conformidad de una implementación frente al estándar son claves para posibilitar la adopción masiva de una tecnología [33].

Dos formas diferentes pero complementarias de realizar pruebas de una tecnología de comunicación son los tests de conformidad y los de interoperabilidad. En los primeros se verifica que una implementación particular cumple satisfactoriamente con lo estandarizado en el protocolo, mientras que en los tests de interoperabilidad se demuestra que dos implementaciones pueden comunicarse correctamente [13].

La principal diferencia entre los mencionados enfoques radica en que los tests de interoperabilidad se basan en aspectos funcionales desde el punto de vista de un usuario, probando dos implementaciones reales de una tecnología. Por otra parte, los tests de conformidad mantienen un mayor control sobre el contenido y secuencia de mensajes al nivel del protocolo implementado. Hay que destacar que los tests de conformidad por sí solos no pueden asegurar la interoperabilidad y que no garantizan la ausencia de errores, simplemente sirven para aumentar la confianza de que la implementación se ha realizado correctamente [24].

La combinación de ambos enfoques, con pruebas de interoperabilidad y conformidad, permite mostrar que implementaciones de dos fabricantes diferentes pueden comunicarse utilizando un protocolo estandarizado.

El proceso antes de que un producto esté listo para lanzarse al mercado utilizando una tecnología de comunicación probada implica: el desarrollo de estándares, definición de tests de validación y certificación. El proceso de certificación, dependiendo de la tecnología, puede implicar tests de conformidad, interoperabilidad o una combinación de ambos. En el caso de LoRaWAN, la LoRa Alliance [1] define un proceso con un conjunto de pruebas que una vez aprobadas permite a un fabricante anunciar su producto como compatible con el estándar.

Contar con herramientas que permitan a los desarrolladores comprobar que su implementación del protocolo es correcta antes de someterse al proceso oficial de certificación resulta fundamental para disminuir costos y acortar los tiempos de desarrollo.

[1] `https://lora-alliance.org/about-lora-alliance`

1.3. Proyecto F-Interop

Para realizar pruebas de conformidad e interoperabilidad de dispositivos se realizan conferencias y eventos a los que concurren diferentes desarrolladores para poner a prueba sus implementaciones [26]. Las principales desventajas del mencionado enfoque son los costos asociados, a veces difíciles de cubrir por pequeñas empresas, y el tiempo añadido en el desarrollo debido a que se debe esperar hasta el siguiente evento para realizar pruebas [33].

F-Interop es un proyecto de investigación Europeo de tres años de duración, finalizado en 2018, y financiado en el marco del programa de innovación Horizon 2020 [2]. El objetivo del proyecto F-Interop fue facilitar la realización de pruebas de forma remota, interconectando algunas plataformas de tests existentes (e.g. Fed4FIRE [3], OneLab [4], IoT Lab [5]). Con la colaboración de diferentes organizaciones de estandarización se fomenta la participación de la industria en la adopción de estándares [3].

F-Interop desarrolló una plataforma extensible de pruebas en línea para tecnologías de IoT, que reduce los costos y tiempos dedicados a tests para:

- Conformidad.
- Interoperabilidad,
- Calidad de Servicio (QoS, Quality of Service).
- Calidad de Experiencia (QoE, Quality of Experience).
- Eficiencia Energética.

Adicionalmente, F-Interop realizó llamados abiertos para extensión de su plataforma agregando módulos de tests. En el marco de estos llamados, desde la Universidad Oberta de Catalunya el Dr. Xavier Vilajosana, co-director de esta tesis, presentó el proyecto F-LoRa como una extensión de F-Interop para realizar pruebas de conformidad de LoRaWAN. El presente trabajo de tesis incluye entre sus objetivos el desarrollo realizado durante un año e integrado a la plataforma F-Interop en Octubre de 2018, publicado además en la conferencia AdHoc-Now 2018 en Saint Malo, Francia [25].

El diseño de F-Interop plantea una plataforma modular y escalable a la que se le pueden añadir nuevos servicios como extensiones con tests de diferentes tecnologías. Cuenta con un conjunto de componentes centrales que se encargan de manejar la sesiones, coordinar el despliegue de los tests y almacenar los resultados, y contempla diferentes herramientas de test (Testing Tools) que implementan los tests específicos de cada tecnología. Para realizar la conexión entre el dispositivo que se desea probar y la plataforma de tests se añade Agentes de usuario (Agent) que se encargan de intercambiar datos con la plataforma. La Fig. 1.1 muestra la arquitectura de F-Interop [25].

[2] `https://ec.europa.eu/programmes/horizon2020/en/what-horizon-2020`
[3] `https://www.fed4fire.eu/the-project/`
[4] `https://onelab.eu/`
[5] `https://www.iotlab.com/en#about`

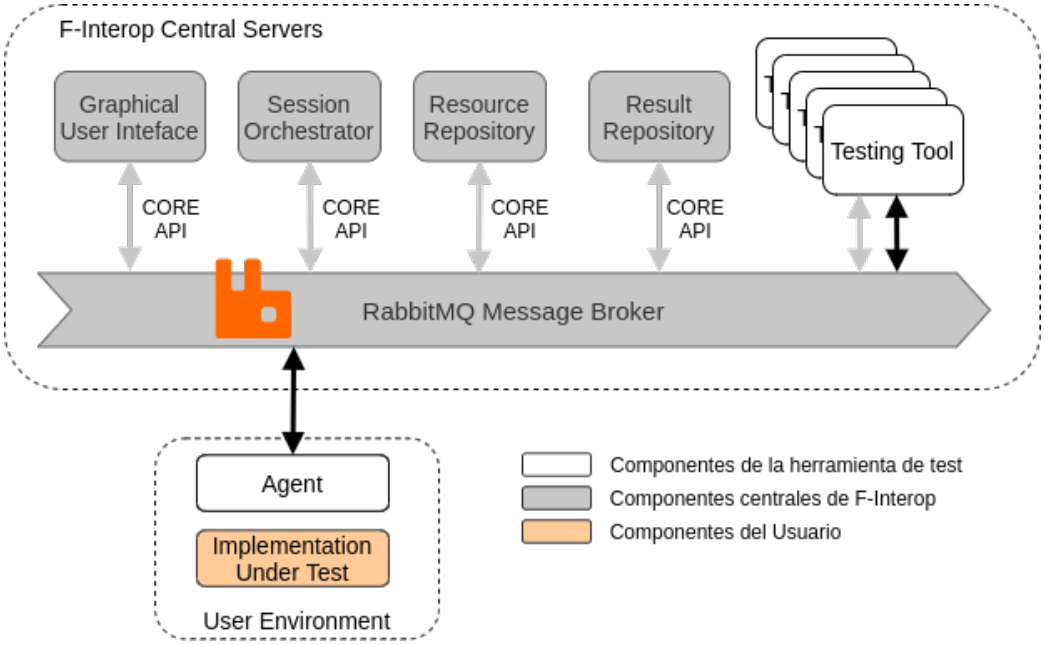

Figura 1.1: Arquitectura de F-Interop, mostrando integración de herramientas de tests al core central.

De acuerdo a los objetivos del trabajo de tesis, tal como se describe en la Subsección 1.4.1, el diseño de la plataforma de tests para LoRaWAN debe poder integrarse en esta arquitectura, además de permitir desplegarse de forma local y ejecutarse sin dependencias externas.

1.4. Objetivos de la Tesis

El objetivo de este trabajo de tesis de maestría es el desarrollo de una plataforma abierta de tests de conformidad para LoRaWAN. Esta plataforma debe permitir conectar un dispositivo de bajo consumo que implementa LoRaWAN y mediante el intercambio de mensajes comprobar que no se detectan errores en la implementación del protocolo.

Se busca proporcionar una herramienta para verificar, en etapas tempranas del desarrollo, que podrán completarse satisfactoriamente las pruebas de certificación de LoRaWAN. Adicionalmente, se espera que esta plataforma sea un recurso educativo para comprender el funcionamiento de LoRaWAN, pudiendo extenderse con nuevos tests que cubran otras funcionalidades o características de diferentes regiones.

El diseño de la plataforma debe permitir la ejecución local y también la integración en otros entornos de pruebas. En particular, se buscó integrar la plataforma desarrollada dentro del proyecto F-Interop, que se describe en la Sección 1.3, siendo este el disparador para la definición de esta tesis de maestría.

1.4.1. Objetivos Específicos

Durante el desarrollo de la plataforma se buscó cumplir los siguientes objetivos:

- Comprender en profundidad el protocolo LoRaWAN, documentando su funcionamiento en forma de que pueda ser utilizado como referencia.
- Revisar conceptos sobre el diseño, implementación y despliegue de un proyecto de software, y aplicarlos para obtener un diseño escalable y fácil de mantener.
- Facilitar la integración de la plataforma diseñada dentro de otros entornos de pruebas. En particular, la implementación debe integrarse con el proyecto F-Interop.
- Obtener una solución que le permita a un usuario desplegar localmente la plataforma para probar un dispositivo sin depender de conexiones a otros entornos de prueba.
- Diseñar la plataforma contemplando la facilidad de extenderla con nuevos tests.

1.4.2. Alcance y Limitaciones

- Los tests implementados están basados en la versión de LoRaWAN 1.0.3.
- Solamente se prueban características de dispositivos de Clase A.
- Se trabajó con la región europea EU 863-870.
- Se implementan tests para cubrir al menos las siguientes características de LoRaWAN:
  - Proceso de activación de los dispositivos.
  - Intercambio de comandos de configuración (MAC).
  - Configuración de los parámetros de Downlink.
  - Mecanismos de encriptación de datos.
  - Cálculos del código de control de integridad de los mensajes: Message Integrity Code (MIC).
- Se proporciona una interfaz de usuario básica para interactuar con la plataforma cuando esta se despliega en forma local.
- El despliegue local no guarda información de sesión de tests y los resultados son simplemente desplegados en forma de logs y a través de la interfaz de usuario.

1.4.3. Publicación del Código Fuente

Todo el código desarrollado y documentación básica sobre la arquitectura están disponibles en un repositorio público de GitHub [6]. En caso de que el lector lo desee, puede resultar de utilidad seguir el código a medida que se avanza en el texto, especialmente al leer el Capítulo 4.

El mencionado repositorio contiene un fichero Makefile que permite realizar el despliegue y ejecución de las sesiones de test, una guia rápida de ejecución está disponible en el Apéndice B. Además de los ficheros correspondietes al despliegue y un directorio con la definición de las imágenes de los servicios (*docker_images*), siguiendo los criterios detallados en el Capítulo 4 se divide el repositorio en un directiorio con los componentes centrales (*conformance_testing*), otro para la implementación de los tests y componentes específicos de LoRaWAN (*lorawan*) y otro con los elementos dedicados a la interacción con el usuario (*user_interface*).

1.5. Estructura del documento

En este capítulo se explicó el contexto en el que se enmarca LoRaWAN cómo tecnología de comunicación, describiendo la importancia de contar con herramientas para realizar pruebas de conformidad en etapas tempranas del desarrollo de dispositivos. Se especificaron los objetivos, alcance y limitaciones de la plataforma desarrollada y se describieron las características y arquitectura del proyecto F-Interop, entorno de pruebas donde se integra el trabajo realizado.

El resto del documento se estructura de la siguiente forma: el Capítulo 2 hace un estudio de las características de LoRaWAN, mostrando la arquitectura, formato de mensajes y mecanismos de optimización de la red definidos en el protocolo. En el Capítulo 3 se abordan aspectos del diseño de la plataforma de tests para LoRaWAN, explicando la arquitectura elegida para lograr los objetivos propuestos sin entrar en detalles de implementación. Este último aspecto es cubierto en el Capítulo 4, donde se fundamenta la elección de tecnologías utilizadas.

Finalmente, en el Capítulo 5 se incluyen las pruebas y resultados obtenidos antes de mostrar las conclusiones generales del trabajo en el Capítulo 6.

[6] `https://github.com/pablomodernell/lorawan_conformance_testing`

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

LoRaWAN

En este capítulo se realiza un resumen de las principales carácteristicas de LoRaWAN, explicando brevemente sus ventajas y limitaciones frente a otras tecnologías de comunicación y presentando la arquitectura y los componentes definidos por el estándar. Se realiza además una descripción de la capa física para comprender cómo las características de esta son utilizadas por los mecanismos definidos por la capa MAC para optimizar el desempeño de la red.

2.1. Capa Física, LoRa

LoRaWAN define el protocolo de comunicación y una arquitectura de red sobre enlaces que utilizan una capa física LoRa. La definición de estos dos elementos resultan clave e impactan directamente en la duración de las baterías de los dispositivos, la capacidad de la red y la calidad de servicio ofrecida por esta [29].

El largo alcance proporcionado por LoRa permite implementar topologías de tipo estrella, con un concentrador central al que se conectan cientos de dispositivos en una región geográfica que podría llegar a cubrir gran parte de una ciudad. Este tipo de topología de red reduce la complejidad del sistema si se la compara con la de uno que implementa una topología de tipo mesh. Las redes de tipo mesh son necesarias cuando se utilizan tecnologías de corto alcance y pueden presentar algunas ventajas como por ejemplo la capacidad de recuperarse y configurar rutas alternativas en caso de que un dispositivo falle. Por otra parte, requieren un costo de infraestructura elevado si se quieren cubrir grandes regiones debido a la gran cantidad de dispositivos que se necesitan.

Una comparación de las topologías de tipo estrella y mesh se puede observar en la Fig. 2.1, donde se muestra un escenario en el que debido al corto alcance proporcionado por la capa física es necesario agregar más nodos solamente para que actúen como repetidores. Además del costo adicional en infraestructura, las topologías de tipo mesh también requieren un mayor consumo de energía ya que los dispositivos deben reenviar mensajes de otros nodos además de manejar sus propios mensajes [29].

El esquema de modulación de LoRa, que es propietario de Semtech y cerrado,

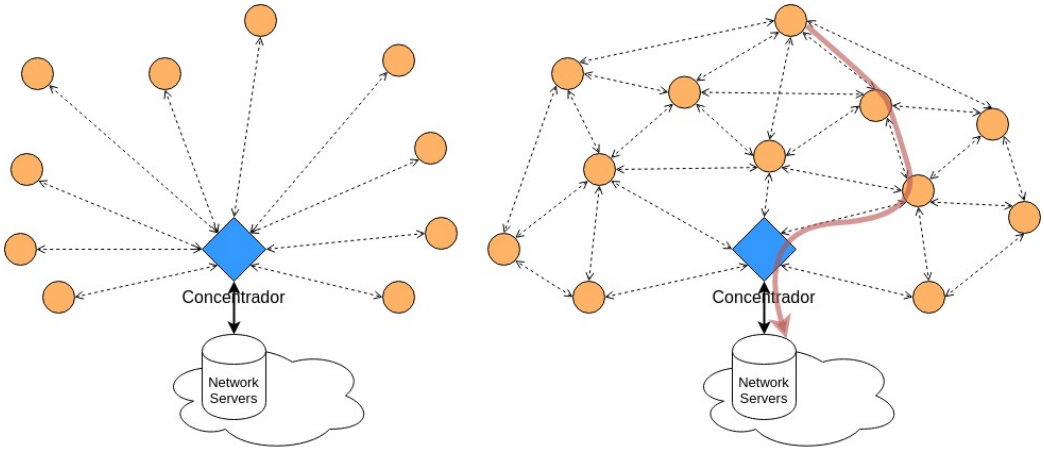

Figura 2.1: Comparación de topologías de tipo estrella y mesh.

está basado en la tecnología Chirp Spread Spectrum (CSS), utiliza un ancho de banda fijo de canal y proporciona alta sensibilidad en comunicaciones a tasas de transmisión de datos bajas.

Mediante la utilización de diferentes Spreading Factor (SF) ortogonales es posible optimizar la red adaptando el Data Rate (DR) según las necesidades. La ortogonalidad de los SF significa que múltiples flujos de datos pueden ser correctamente detectados y recibidos en un mismo canal al mismo tiempo. Por lo tanto, la gestión de los SF y canales utilizados se vuelve relevante y esta tarea es asumida por LoRaWAN, que optimiza los parámetros de comunicación utilizados por los dispositivos [17].

La combinación del SF y el Bandwidth/Ancho de banda (BW) utilizado determina la duración de cada símbolo y los niveles de señal necesarios para detectar la transmisión [29]. Es decir, para un ancho de banda fijo, utilizar mayores SFs implica reducir el tiempo requerido para enviar un mensaje pero también aumenta el nivel de señal que se requiere para para decodificar la señal. LoRaWAN trabaja con los mencionados DR para referirse a las distintas combinaciones de SF y BW, donde un DR más bajo corresponde a menor tasa de transferencia de datos pero a comunicaciones más robustas. La correspondencia de cada valor válido de DR con un SF y BW está está especificado en el documento de parámetros regionales, y varía por región. Por ejemplo, la Tabla 2.1 muestra esta información para la región Europea [2].

La Fig. 2.2 muestra cómo LoRaWAN proporciona servicios a la capa de Aplicación basándose en el sistema de modulación de radio de la capa física ofrecido por LoRa [30]. La capa física además estará configurada para funcionar en determinada banda de frecuencias del espectro ISM (Industrial, Scientific and Medical), dependiendo de las regulaciones de radio que se apliquen en la región donde funcione el sistema [2] [29].

Tal como será explicado en la Sección 2.4, que describe la arquitectura definida por LoRaWAN, los dispositivos se comunican de forma bidireccional con una red central que los conecta con aplicaciones. Existen diferencias en cómo se utiliza

Data Rate	Configración (SF/BW)	Tasa de bits por segundo [bit/seg]
0	SF12 / 125 kHz	250
1	SF11 / 125 kHz	440
2	SF10 / 125 kHz	980
3	SF9 / 125 kHz	1760
4	SF8 / 125 kHz	3125
5	SF7 / 125 kHz	5470
6	SF7 / 250 kHz	11000

Tabla 2.1: Configuraciones válidas de Data Rate en LoRaWAN usando LoRA en la región Europea (EU868).

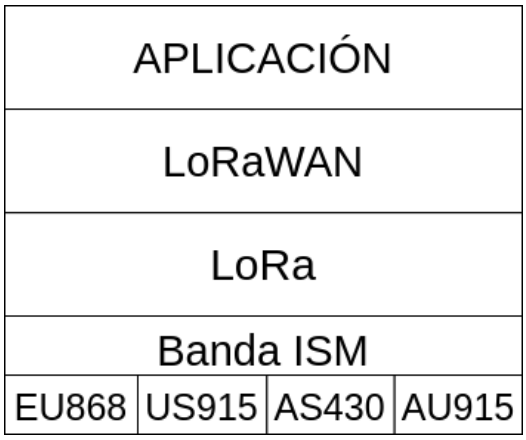

Figura 2.2: Diagrama de capas de comunicación implementadas por un dispositivo.

la capa física según la región de funcionamiento del sistema para adaptarse a las distintas regulaciones de radio.

Las mencionadas diferencias entre regiones son por ejemplo en el ámbito de la potencia máxima de transmisión autorizada, los canales y anchos de banda a utilizarse y la existencia de ciclos de trabajo Duty Cycle (DC) o límites en el tiempo para la transmisión de mensajes en cada uno de los canales (Dwell Time). En la región de América del Norte se utiliza la banda de frecuencias US 915 donde se definen 80 canales con anchos de banda de 125 o 500 kHz y se limita con un tiempo máximo de duración para las transmisiones (Dwell Time) de 400 ms. Por otra parte en Europa, donde se utiliza la banda de frecuencias EU 868 se definen 10 canales de comunicación en donde se utiliza un ancho de banda de 125 kHz o 250 kHz para Downlink y 125 kHz para Uplink. En la región europea se definen máximos de DC de 0,1 %, 1 % y 10 % en tres diferentes sub bandas, limitando de esta forma el porcentaje máximo de tiempo de utilización de los canales que comprenden cada una de las sub bandas. Esto significa, para plantearlo a modo ilustrativo, que un

dispositivo solo podrá transmitir utilizando las frecuencias correspondientes a las sub-bandas limitadas a un 10 % de DC un total de 6 minutos en cada hora [11] [28].

En Uruguay se utiliza la región AU915 con frecuencias en el rango de 915 a 928 MHz. Se definen 64 canales de 125 kHz, y 8 de 500 kHz, de ancho de banda para el Uplink y 8 canales de 500 kHz de ancho de banda para el Downlink [2].

LoRa ofrece una implementación de CSS que permite reducir la complejidad en el diseño de los receptores y es adecuada para dispositivos de bajo consumo, logrando de esta forma un gran alcance en las comunicaciones con chips de bajo coste de fabricación. Puede además adaptarse mediantes simples configuraciones a distintos modos de operación y es un sistema robusto ante interferencias y a los efectos de desvanecimiento por trayectos múltiples (multipath fading). Estas ventajas del sistema de modulación pueden ser traducidas en un incremento de hasta cuatro veces en el alcance de las comunicaciones [29].

2.2. Estandarización y LoRa Alliance

Tal como se mencionó en la sección anterior, el protocolo y esquema de modulación definido por LoRa es cerrado y propietario de la empresa Semtech [1]. A pesar de esto, en un esfuerzo por estandarizar las LPWAN, facilitando la adopción de estas tecnologías, Semtech impulsó la creación de la LoRaAlliance [2] para trabajar en la definición del estándar abierto LoRaWAN. Además de LoRa, el estándar LoRaWAN puede adaptarse a trabajar sobre otra capa física, en particular se contempla su utilización sobre modulación FSK (Frequency Shift Keying) [2] [29].

La LoRa Alliance trabaja desde Marzo de 2015 y está integrada por más de 500 miembros entre los que se encuentran IBM, Cisco, HP, Foxconn y Semtech [4]. Además de definir un estándar abierto para la capa MAC, esta organización define un programa de certificación para asegurar la interoperabilidad entre productos que utilizan el estándar. Todo el ecosistema definido alrededor de LoRaWAN intenta posicionarla como la tecnología más adoptada dentro de las LPWAN [18] [6].

El resultado del trabajo de la LoRa Alliance se refleja en los diferentes documentos que publica:

- Especificación de LoRaWAN: indicando las especificaciones técnicas generales básicas del protocolo [30].
- Parámetros Regionales: con información sobre las particularidades del protocolo en cada una de las regiones (hay un documento por región) [2].
- Procedimientos de Certificación: especificando las pruebas de certificación a las que debe someterse un dispositivo para poder decir que cumple con el protocolo. Estos documentos están accesibles para miembros de la LoRa Alliance, requerimiento para poder realizar las pruebas de certificación.

[1] `https://www.semtech.com/lora/`

[2] `https://lora-alliance.org/about-lora-alliance`

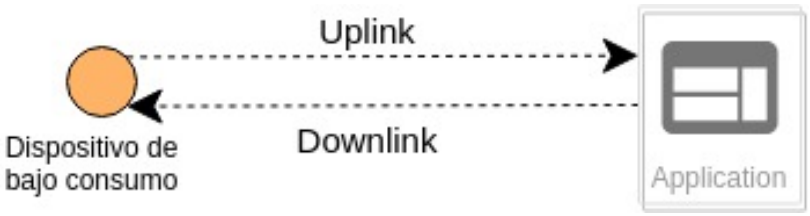

Figura 2.3: Esquema de comunicación bi-direccional entre un dispositivo de bajo consumo y una aplicación.

- Interfaces de Back End: documento indicando la división de la red en diferentes componentes, y los protocolos de comunicación entre estos, con el objetivo de facilitar la interconexión de proveedores de red.

Cada versión de la especificación de LoRaWAN introduce cambios y mejoras en el protocolo, el formato de los mensajes, los componentes que intervienen y las interfaces que los interconectan. El mayor cambio en cuanto a la definición de la red LoRaWAN se da a partir de la versión 1.1 del estándar, que está fuera del alcance de este trabajo de tesis basado en la versión 1.0.3. Además de realizar algunas puntualizaciones donde resulta relevante aclarar diferencias entre estas versiones, cómo por ejemplo en el nombre de algunos identificadores de sesión, en la Sección 2.6 se hace un breve resumen de las principales mejoras introducidas en LoRaWAN 1.1.

Adicionalmente a los documentos con especificaciones y a los programas para certificación de dispositivos, la LoRa Alliance se encarga de publicar recomendaciones técnicas para los fabricantes apoyando una implementación óptima de los mecanismos de seguridad [5].

A continuación se analiza la especificación general de LoRaWAN en base a lo definido en los mencionados documentos [2] [30], haciendo especial énfasis en los aspectos del protocolo para los que se han implementado tests de conformidad en el presente trabajo de tesis.

2.3. Comunicación bi-direccional con LoRaWAN

La comunicación es bi-direccional entre dispositivos de bajo consumo y aplicaciones que procesan los datos recibidos. En este intercambio de información se hace referencia a los mensajes que se dirigen desde los dispositivos hacia la aplicación cómo mensajes de Uplink y a los mensajes que van desde la aplicación hacia los dispositivos como mensajes de Downlink. En la Fig. 2.3 se muestra un esquema con un dispositivo de bajo consumo intercambiando mensajes con una aplicación. En un escenario típico se espera que una multitud de dispositivos envíen datos a la misma aplicación, esto por ejemplo podría ser un conjunto de sensores de parking que se comunican con la aplicación que gestiona la movilidad en una ciudad.

De acuerdo a la especificación, los dispositivos en una red LoRaWAN pueden utilizar cualquier canal para transmitir un mensaje, utilizando el DR y SF que elijan, siempre que se cumpla con las siguientes condiciones:

- la selección del canal para cada transmisión debe ser pseudo-aleatoria para que el sistema sea más robusto a interferencia,
- se debe cumplir la regulación respecto al DC máximo para las transmisiones en la región donde se está operando,
- los tiempos máximos de transmisión (Dwell Time) en la sub-banda utilizada deben respetar lo establecido para la región.

En lo que respecta a los mensajes de Downlink, se definen tres clases de dispositivos:

- Clase A: pensada para dispositivos que necesiten permanecer en modo de bajo consumo gran parte del tiempo, prioriza el tráfico de Uplink y solo permite el envío de mensajes de Downlink en dos ventanas que se abren un tiempo predeterminado luego de finalizado el envío de un mensaje de Uplink. La implementación de este modo de funcionamiento es obligatoria para todos los dispositivos.
- Clase B: se define para aplicaciones que requieran el envío de mensajes de Downlink con una menor latencia, donde no sea aceptable esperar a que el dispositivo envíe un mensaje de Uplink para poder comunicarse con él. En este modo de funcionamiento una señal de Beacon enviada desde los Gateways se utiliza para sincronizar los nodos y hacerlos abrir ventanas de recepción de Downlink en intervalos de tiempo conocidos. Esta clase de dispositivos buscan un balance entre una latencia acotada por el intervalo de tiempo entre los mensajes de Beacon y un consumo de energía aceptable.
- Clase C: para dispositivos que no tengan restricciones en el consumo de energía se define este modo de funcionamiento de escucha continua, logrando de esta forma minimizar la latencia cuando la red necesita enviar un mensaje de Downlink a un dispositivo.

De acuerdo al funcionamiento definido para la clase A, cada vez que un dispositivo termina de enviar un mensaje de Uplink queda a la espera de recibir un mensaje de Downlink que la red pueda tener agendado para él. Para recibir este mensaje de Downlink se enciende la radio durante una ventana de recepción *RX1* que comienza luego de un período de tiempo *RECEIVE_DELAY1*. Este período de tiempo es negociado entre el dispositivo y la red durante el inicio de sesión o mediante el intercambio de comandos de configuración. Sólo en caso de que no se reciba ningún mensaje de Downlink en esta ventana de tiempo, una segunda ventana de Downlink *RX2* es abierta un segundo después de iniciada la primera.

Un diagrama de tiempo explicando la secuencia de eventos entre la transmisión de un mensaje y la apertura de las ventanas de Downlink se muestra en la Fig. 2.4, donde se indica la relación entre los parámetros de tiempo definidos y se aprecia un tiempo fijo de un segundo entre los instantes de apertura de las ventanas de Downlink.

Los valores por defecto para los parámetros de comunicación utilizados en las ventanas de Downlink *RX1* y *RX2* pueden variar en cada una de las regiones

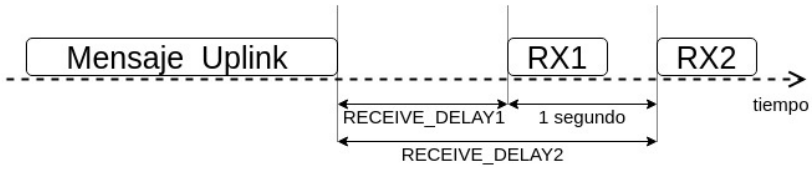

Figura 2.4: Diagrama de tiempo de las ventanas de Downlink.

donde se implementa LoRaWAN, pero actualmente todas las regiones proponen valores por defecto de 1 segundo para *RECEIVE_DELAY1*, que implica que *RECEIVE_DELAY2* será de 2 segundos. El test td_lorawan_fun_02, detallado en la Subsección A.2.2 del Apéndice A, prueba que la tolerancia a errores en el tiempo de inicio de estas ventanas de recepción cumpla la especificación de LoRaWAN.

Además de especificar el momento en el que que el dispositivo estará escuchando estas transmisiones, otro aspecto a configurar de las ventanas de Downlink es qué canal utilizar y el DR para las transmisiones. Para la ventana *RX1* se utilizan parámetros en función de los utilizados en la transmisión de Uplink, con algunas variaciones en cada región. Por ejemplo, en la región *EU 868* se espera que el Downlink en *RX1* utilice el mismo canal empleado en el último Uplink y un DR que tiene un offset configurable. Para la ventana *RX2* se utilizan parámetros fijos de frecuencia y DR, por defecto en valores: 868,525 MHz y DR 0 (el más bajo, que implica comunicaciones más robustas). La correcta implementación de estas configuraciones al momento de iniciar sesión se pone a prueba en el test td_lorawan_act_03, que se describe en la Subsección A.1.3 del Apéndice A.

Este trabajo se centró en la especificación de LoRaWAN versión 1.0.3 para la Clase A, por lo que solo se cubren aspectos que son obligatorios de implementar en todos los dispositivos. Las siguientes secciones resumen la especificación del protocolo, pasando por la definición de la arquitectura, formato de los mensajes, mecanismos de seguridad implementados y el intercambio de mensajes de comandos utilizados para configurar la red (comandos MAC).

2.4. Arquitectura de red LoRaWAN

LoRaWAN define una arquitectura de red que utiliza la topología de tipo estrella permitida por la capa física LoRa, y define una estrella de estrellas con un servidor de red (Network Server) recibiendo los datos de varios concentradores (también denominados Gateways en el contexto de LoRaWAN). Los dispositivos no están asociados a un Concentrador/Gateway en particular, sus mensajes pueden ser recibidos por cualquier Gateway de la red que se encargará de enviarlos hacia el servidor de red central.

La Fig. 2.5 presenta un diagrama con la arquitectura donde se ven los distintos elementos involucrados: dispositivos, Gateways, Network Server central y servidores de aplicación que reciben los datos enviados por los dispositivos. La comunicación entre los dispositivos y cada concentrador es a través de un enlace LoRa. Por otra parte, la conexión entre los Gateways y el Network Server, así como

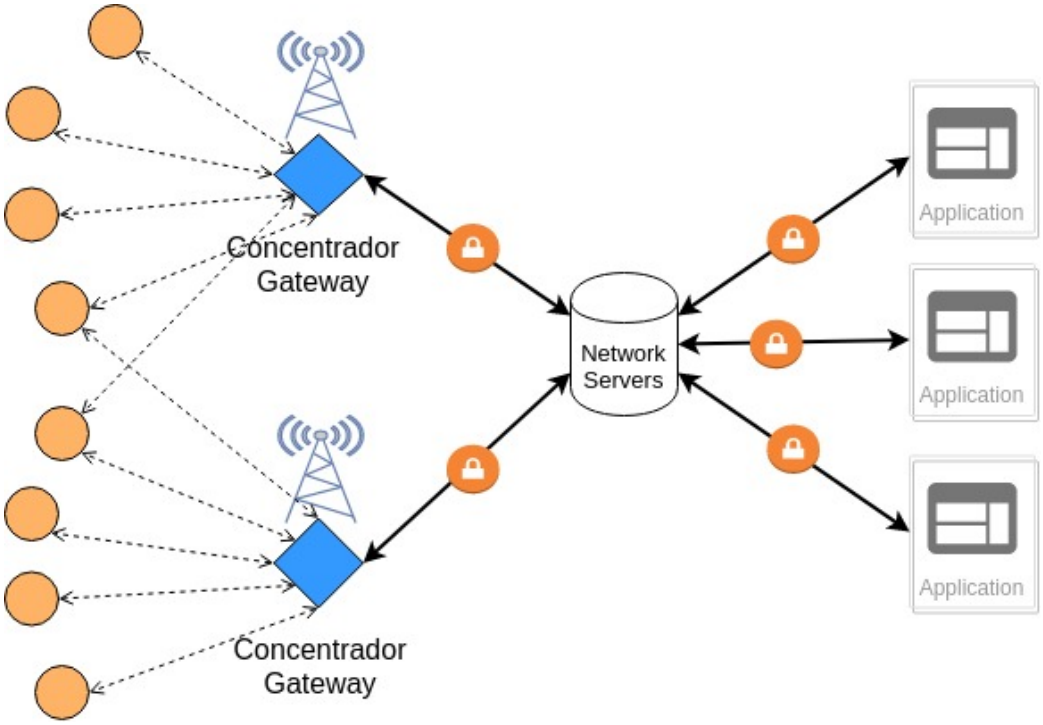

Figura 2.5: Arquitectura de una red LoRaWAN.

la de este último con los distintos servidores de aplicación es mediante enlaces IP confiables y seguros que pueden utilizar Ethernet, Wi-Fi o redes celulares.

Los dispositivos LoRaWAN establecen comunicación bidireccional con aplicaciones que consumen y procesan los datos que estos envían utilizando la red compuesta por el Network Server y los Gateways. En esta comunicación bidireccional se priorizan los mensajes de Uplink frente a los mensajes de Downlink, debido a que los dispositivos se comunicarán de forma asíncrona cuando tienen un dato para enviar y estarán en modo de bajo consumo apagando sus receptores el resto del tiempo.

Los Concentradores o Gateways LoRa simplemente decodifican todos los mensajes recibidos y los envían al Servidor de Red (Network Server) que tengan configurado, sin analizar el contenido de estos. De hecho, debido a los mecanismos de seguridad implementados en LoRaWAN, los Gateways no cuentan con las claves de encriptación necesarias para autenticar ni para ver el contenido de los mensajes.

La inteligencia de la red está concentrada en el Network Server, que desempeña las siguientes funciones principales:

- Manejo del intercambio de mensajes para inicio de sesión, verificando luego los identificadores asignados a los dispositivos.
- Envío de los mensajes recibidos hacia la aplicación correspondiente.
- Verificación de integridad de los mensajes.

- Filtrado de mensajes duplicados (que pueden llegar desde más de un Gateway).
- Agendado de los mensajes de Downlink, decidiendo el momento y el Gateway particular que debe enviar un mensaje hacia un dispositivo.
- Gestión de los parámetros de comunicación (frecuencias, SF, etc.) para cada uno de los dispositivos conectados.

La gestión de los parámetros de la capa física LoRa de los dispositivos implica elegir las frecuencias y DR para maximizar a la vez la capacidad de la red y la duración de las baterías. Los DR altos, que utilizan menores SF, reducen el tiempo en el aire de los mensajes y por lo tanto ocupan menos el canal y ahorran batería a los dispositivos. Por otra parte, en casos donde las distancias sean grandes o exista presencia de interferencia un SF más alto puede ser necesario para hacer la transmisión más robusta y evitar la pérdida de mensajes.

Si bien es el Servidor de Red LoRaWAN quien gestiona los canales de comunicación, los Gateways LoRa asociados a la red deben soportar los canales negociados entre los dispositivos y la red para que la comunicación pueda realizarse correctamente.

Los servidores de aplicación implementan el procesamiento de los datos proveniente de los dispositivos de bajo consumo, estableciendo la comunicación a nivel de capa de aplicación. Una vez que el dispositivo cuenta con los identificadores y claves necesarios para establecer la comunicación con el Servidor de Aplicación asignado decimos que este dispositivo ha iniciado una sesión.

2.4.1. Seguridad

Como cualquier sistema de comunicación inalámbrico, una red LoRaWAN está expuesta a distintos tipos de ataques que podrían afectar su funcionamiento (e.g. denegación de servicio) y comprometer la integridad y confidencialidad de los mensajes intercambiados. Deben por lo tanto implementarse mecanismos que garanticen la autenticación mutua de los dispositivos y la aplicación, minimizando el riesgo de suplantación de identidad y ataques de replay. Además, resulta fundamental contar con una correcta protección de integridad y encriptación de los mensajes.

La seguridad en una red LoRaWAN utiliza el Advanced Encryption Standard con claves simétricas de 128 bits (AES-128), utilizado para garantizar las características de autenticación, integridad y confidencialidad de los datos mencionadas anteriormente.

Para la confidencialidad de los mensajes intercambiados entre dispositivos y aplciaciones se utiliza la encriptación simétrica definida en el modo de operación AES-CCM del estándar IEEE 802.15.4-2011. El dispositivo de bajo consumo y la aplicación deben compartir una clave de encriptación *Application Session Key (AppSKey)* válida para cada sesión de comunicación. Por otra parte, para implementar las comprobaciones de integridad de los mensajes se utiliza el modo de operación AES-CMAC definido en el RFC4493, con otra clave de sesión a nivel

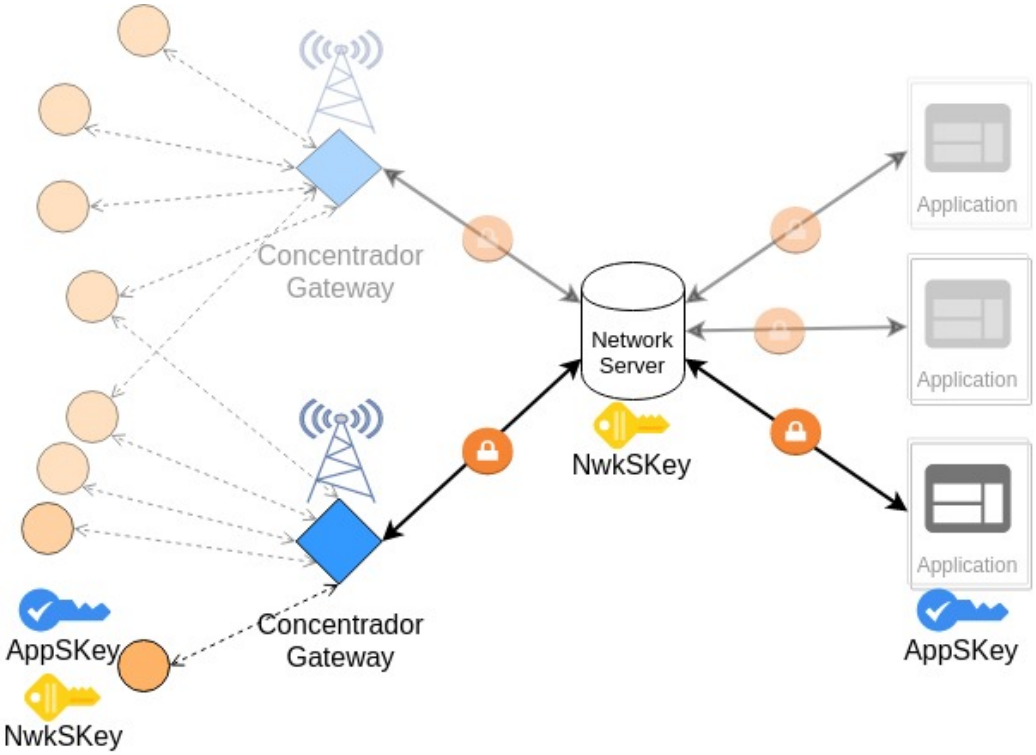

Figura 2.6: Claves de un dispositivo en un red LoRaWAN.

de red denominada *Network Session Key (NwkSKey)*. En la Fig. 2.6 se muestra como una vez iniciada la sesión en la red LoRaWAN el dispositivo cuenta con las dos claves de sesión mencionadas, estando la *AppSKey* compartida solamente con el servidor que maneja la aplicación y la *NwkSKey* compartida con el servidor de red (Network Server).

Estas claves de sesión podrán renovarse dinámicamente (modo recomendado de uso) o permanecer fijas durante toda la vida útil del dispositivo, dependiendo del modo de funcionamiento del mismo. La importancia de la distinción entre estas dos claves radica en que el Network Server, que puede estar gestionado por un proveedor de red, solamente cuenta con la clave que le permite comprobar la integridad de los mensajes y, como se explicará más adelante, encriptar mensajes de comandos.

Los tests td_lorawan_sec_01 y td_lorawan_sec_02, detallados en la Subsección A.3.1 y Subsección A.3.2 del Apéndice A, ponen a prueba la implementación de los mecanismos de encriptación y de generación de códigos de integridad. En las siguientes secciones se muestra cómo se definen los procesos de inicio de sesión para la generación de las mencionadas claves, para luego explicar el formato de los mensajes y las formas definidas para el intercambio de comandos de configuración.

2.4.2. Inicio de sesión y activación de dispositivos

La activación de un dispositivo implica la generación y distribución de las claves de sesión y la asignación de identificadores. Estos son los identificadores y claves definidas para cada sesión de un dispositivo:

- DevAddr: Dirección corta (32 bits) asignada por la red LoRaWAN a cada dispositivo y válida durante la sesión. No tiene por qué ser única y está compuesta por un identificador de la red (*NwkID*) en los 7 bits más significativos, y una dirección de red del dispositivo (*NwkAddr*) formada por los 25 bits restantes.
- AppSKey: Clave de sesión para encriptación del payload de capa de aplicación, compartida entre cada dispositivo y el Servidor de Aplicación.
- NwkSKey: Clave de sesión para la generación del MIC y encriptación de comandos MAC, compartida entre el dispositivo y el Servidor de Red (Network Server).

Cada dispositivo conoce sus claves y dirección corta de red. Además, en cada componente de la red LoRaWAN se conoce:

- en el Servidor de Red,
  - NwkSKey: Clave de sesión compartida con el dispositivo.
  - DevAddr: El servidor debe conocer este parámetro para cada dispositivo, para poder realizar el correcto mapeo con la aplicación luego de verificar la integridad del mensaje (usando la NwkSKey correspondiente a la sesión del dispositivo).
- En el Servidor de Aplicación:
  - AppSKey: Clave de sesión compartida con el dispositivo para encriptación de los datos.

Una vez que estos parámetros están configurados el dispositivo podrá enviar mensajes que serán recibidos por un Gateway LoRa conectado a la red LoRaWAN y reenviados al servidor de Red. El formato definido para estos mensajes se muestra en la Fig. 2.7, donde se detalla el payload de la capa física LoRa PHYPayload entregado al servidor de red (junto con, eventualmente, metadata del mensaje).

Figura 2.7: Formato básico de los mensajes en una red LoRaWAN.

El PHYPayload en una red LoRaWAN comienza siempre con un byte de encabezado de capa MAC, *MAC Header (MHDR)*, que indica el tipo de mensaje.

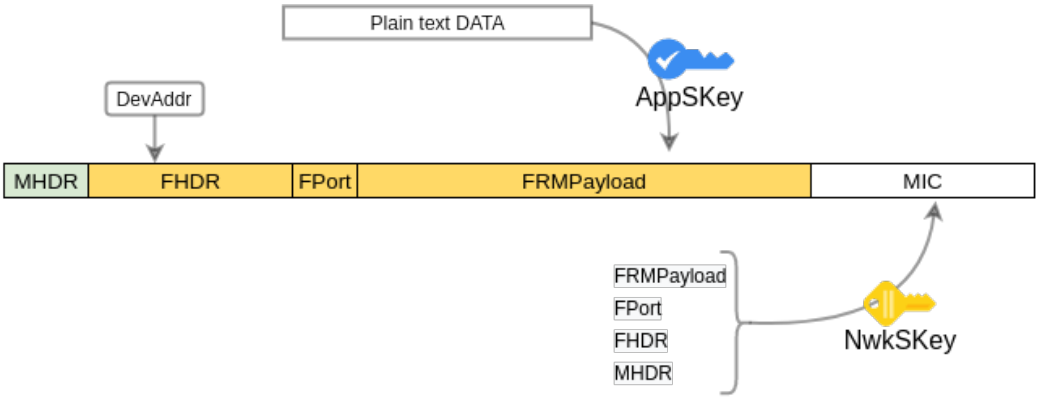

Figura 2.8: Formato definido por LoRaWAN para los mensajes de datos.

La Tabla 2.2 muestra los tipos de mensaje definidos en base a los primeros tres bits del *MHDR* (*MType*). El resto del mensaje se compone por el MACPayload y finaliza con 4 bytes correspondientes al MIC, que es calculado usando la clave de sesión de red (*NwkSKey*). Una vez configurados todos los parámetros necesarios el dispositivo podrá enviar mensajes de datos según lo mostrado esquemáticamente en la Fig. 2.8.

MType bits (MHDR)	Tipo de Mensaje
000	JOIN Request
001	JOIN Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU
111	Propietario

Tabla 2.2: Tipos de mensajes definidos en una red LoRaWAN en base a los primeros 3 bits del MHDR (MType).

El dispositivo genera un mensaje de datos que contiene la *DevAddr* como los primeros 4 bytes del encabezado de trama, *Frame Header (FHDR)*, un byte correspondiente al *Frame Port (FPort)* se utiliza para indicar si el contenido del mensaje son comandos MAC o para identificar diferentes servicios dentro de un mismo Servidor de Aplicación. Finalmente, el dispositivo utiliza la *AppSKey* para encriptar los datos que se desea enviar en el *Frame Payload (FRMPayload)* y la *NwkSKey* para calcular el MIC en base al MHDR y al MACPayload (compuesto por FHDR, FPort y FRMPayload).

Cuando el Servidor de Red recibe el mensaje, dado que conoce la *NwkSKey* puede verificar la integridad y enviarlo a la aplicación correspondiente (en base a la DevAddr). Debido a que no cuenta con la *AppSKey* no es capaz de desencriptar los datos contenidos en el mensaje, esto es particularmente conveniente en casos de que sea un tercero quien opere la red. En la práctica, sin embargo, las implementaciones reales hacen difícil separar los servidores de Aplicación y de Red y esto provoca que no siempre se pueda asegurar que la red no tiene acceso a la clave de aplicación. A partir de la especificación de LoRaWAN V1.1, se establece un diseño de red que facilita la independencia entre un proveedor de red y quien implementa las aplicaciones.

El Servidor de Aplicación recibirá el FRMPayload y podrá desencriptarlo ya que cuenta con la *AppSKey*.

2.4.3. Modos de Activación de Dispositivos

LoRaWAN define dos modos de activación de dispositivos, es decir, dos métodos diferentes para lograr que todos los componentes obtengan las claves e identificadores necesarios para establecer la comunicación.

Activation By Personalization (ABP)

La forma más simple de lograr que un dispositivo esté activado, y con una sesión lista para enviar datos, es mediante la llamada Activation By Personalization (ABP). En este método de activación se configura el dispositivo con claves de sesión y dirección de red (DevAddr) fijas, indicando en los Servidores de Aplicación y de Red que el dispositivo se activa por ABP y proporcionando las claves de sesión, *AppSKey* y *NwkSKey* respectivamente.

Si bien la activación de tipo ABP podría ser conveniente en algunas circunstancias, se recomienda fuertemente utilizarlo solo si no queda otra alternativa, ya que implica riesgos de seguridad por no poder renovar las claves de sesión. Los dispositivos quedan más expuestos a ataques de fuerza bruta para descubrir las claves utilizadas y además se dificulta la definición de métodos que permitan migrar los dispositivos a nuevas redes, ya que se debería contar con una forma segura para mover las claves de una red a otra.

El test td_lorawan_act01, que se describe en la Subsección A.1.1 del Apéndice A, prueba que un dispositivo es capaz de enviar datos una vez configuradas las claves según el procedimiento de ABP.

Over The Air Activation (OTAA)

La alternativa implementada para lograr una mejora en la seguridad y flexibilidad al momento de configurar los dispositivos es la activación de tipo Over The Air Activation (OTAA). En este método de activación la red LoRaWAN tiene un listado de los dispositivos autorizados para comunicarse, y comparte con cada uno de ellos una clave raíz (idealmente una diferente para cada dispositivo). El dispositivo conoce la dirección del servidor con el que quiere autenticarse y en base a la

información compartida se define una secuencia de mensajes, ejemplificada en la Fig. 2.9, que permiten intercambiar claves de sesión:

- JOIN Request, mensaje que un dispositivo envía a la red indicando su identidad y la de la aplicación con la que quiere comunicarse. Este mensaje no está encriptado, por lo que cualquiera puede ver su contenido, pero está firmado para asegurar que proviene desde el dispositivo.
- JOIN Accept, mensaje generado por un servidor encargado de manejar las sesiones y enviado como respuesta hacia el dispositivo. Este mensaje, que está encriptado y solo el dispositivo puede ver su contenido, tiene las claves de sesión y la dirección de red asignada al dispositivo.

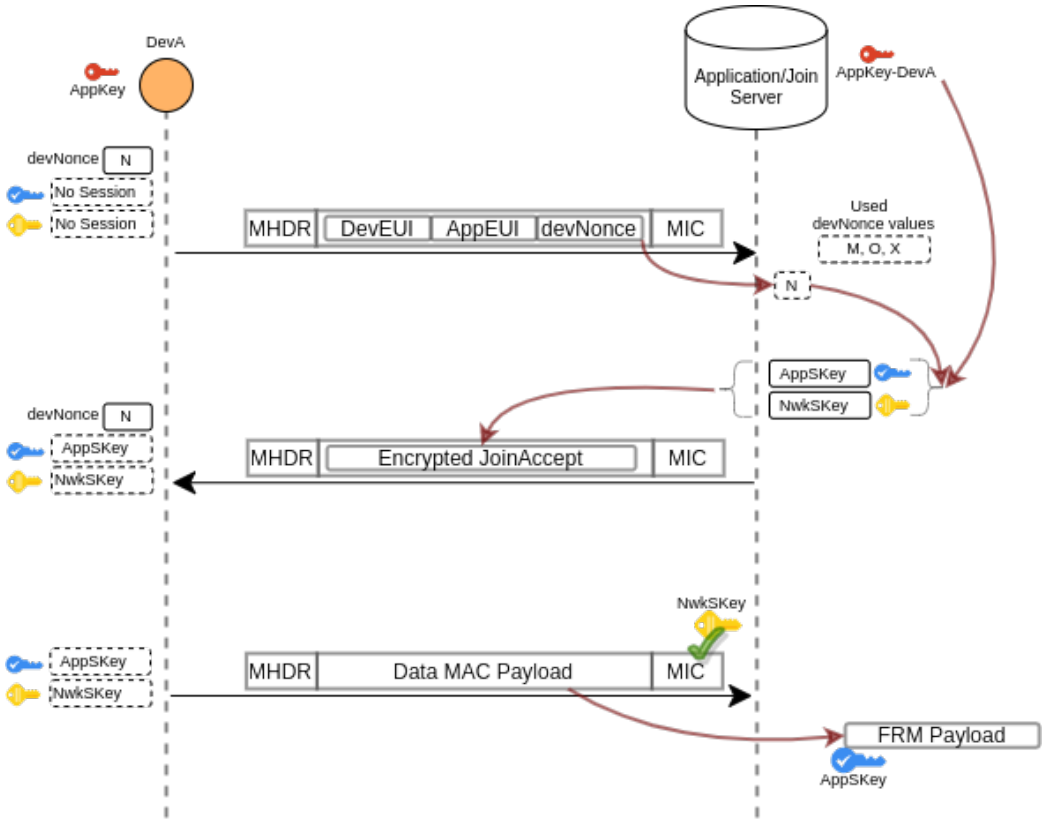

Figura 2.9: Ejemplo de intercambio de mensajes para inicio de sesión con el mecanismo Over The Air Activation.

Para poder iniciar el intercambio de mensajes de JOIN que activan a los dispositivos, estos son los parámetros y claves que deben configurarse en cada uno de los componentes involucrados:

- En dispositivos de bajo consumo:
  - DevEUI: Identificador global único de acuerdo al estándar IEEE EUI64, asignado generalmente por el fabricante.

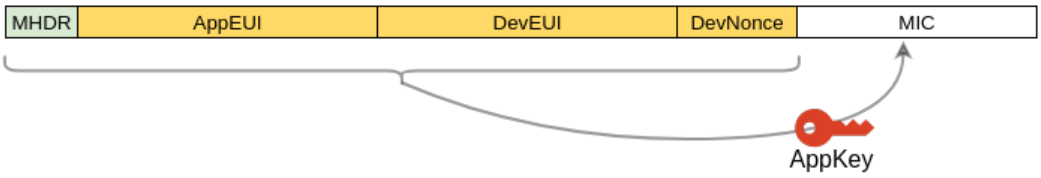

Figura 2.10: Formato de un mensaje de JOIN Request.

  - Application Key (AppKey): Clave raíz (AES-128) utilizada para generar las claves de sesión. Compartida por el dispositivo y el servidor encargado de generar las claves: en las versiones hasta V1.0.3 del protocolo (referencia para este trabajo de tesis) es el Servidor de Aplicación quien cumple este rol, a partir de la versión V1.1 de LoRaWAN se define un JoinServer encargado de almacenar las claves raíz y generar las claves de sesión.
  - AppEUI: Identificador del Servidor de Aplicación, global y único según el estándar IEEE EUI64. A partir de LoRaWAN 1.1 se hace referencia a este identificador como JoinEUI.

- Servidor de Aplicación [3]:
  - Registro con el listado de dispositivos autorizados para iniciar sesión, con su DevEUI y la AppKey correspondiente.
  - AppEUI

Con la información almacenada en el dispositivo, cuando este se enciende es capaz de generar un mensaje que contiene el DevEUI que lo identifica, el AppEUI que identifica a la aplicación con la que se quiere comunicar y un DevNonce de dos bytes, generado aleatoriamente para dificultar los ataques de replay. La Fig. 2.10 muestra el formato definido para los mensajes de JOIN Request, en donde el MHDR indica que el mensaje corresponde a un JOIN Request y el MIC es calculado utilizando la clave raíz *AppKey*.

Cuando la red LoRaWAN recibe un mensaje de JOIN Request, el servidor encargado del proceso de JOIN verifica que:

- el dispositivo que lo originó esté registrado (en base a su DevEUI) y se cuenta por lo tanto con la clave raíz *AppKey*,
- el mensaje está firmado correctamente, comprobando el MIC,
- y que el DevNonce no pertenece a la lista de valores utilizados anteriormente.

Si alguna de las mencionadas verificaciones falla el mensaje es descartado. Los servicios de inicio de sesión de red, además de guardar una lista de valores de

[3] A partir de la versión de LoRaWAN 1.1 es un nuevo componente, el JoinServer, quien se encarga de gestionar las claves.

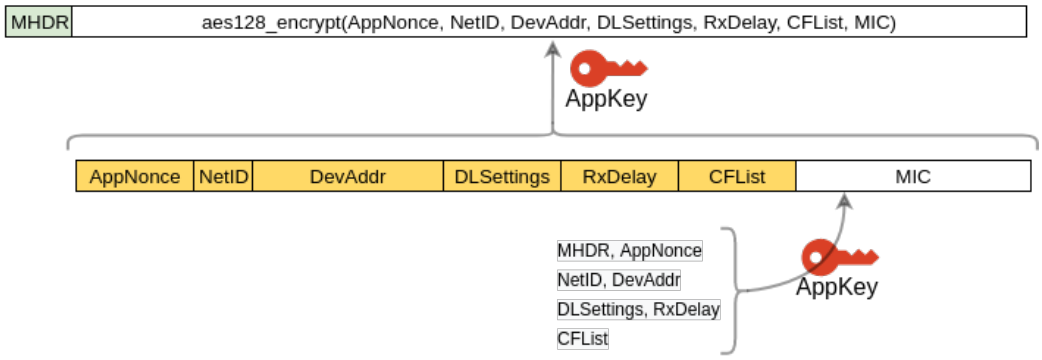

Figura 2.11: Formato de un mensaje de JOIN Accept.

DevNonce utilizados, esperan a recibir un mensaje de datos válido proveniente del dispositivo antes de confirmar la nueva sesión y actualizar las claves. Este procedimiento dificulta ataques de replay donde se pretenda dejar sin servicio a un dispositivo mediante el envío de mensajes de JOIN Request que por ser demasiado antiguos pueden no estar en la lista de los almacenados en el servidor. Como mejora de seguridad, LoRaWAN 1.1 modifica el DevNonce para que sea un contador en vez de un valor aleatorio.

Una vez que el mensaje de JOIN Request es considerado válido el servidor a cargo de las sesiones genera un mensaje de JOIN Accept y lo envía como respuesta. La Fig. 2.11 muestra el formato definido para los mensajes de JOIN Accept, que contiene los parámetros a partir de los cuales se pueden generar las claves de sesión.

El PHYPayload del mensaje consiste en el MHDR que indica que el mensaje es un JOIN Accept, y luego los demás bytes del mensaje (incluido el MIC) están encriptados utilizando la clave raíz *AppKey*. Una particularidad de este proceso es que la operación que se usa en el servidor de la red LoRaWAN es la de AES *Decrypt* y los dispositivos de bajo consumo deben aplicar la operación AES *Encrypt* para obtener los datos del mensaje, de esta forma los dispositivos sólo deben implementar una de estas operaciones (*Encrypt*).

De forma similar a las ventanas de Downlink iniciadas luego de que se transmite un mensaje de datos, el dispositivo estará esperando el mensaje de JOIN Accept en dos ventanas de recepción. La única diferencia entre estas y las ventanas de recepción *Rx1* y *Rx2* descritas en la Sección 2.3 es que para el proceso de JOIN comienzan un tiempo *JOIN_ACCEPT_DELAY1* y *JOIN_ACCEPT_DELAY2* segundos después de terminada la transmisión del mensaje de JOIN Accept. Estos parámetros varían según la región y son por defecto 5 y 6 segundos en la región EU 868.

En cuanto a los parámetros enviados en el mensaje de JOIN Accept:

- AppNonce, son 3 bytes generados aleatoriamente para cada mensaje y usados para derivar las claves de sesión.
- NetID, identificador de red de 3 bytes. Se hace referencia a los 7 bits menos significativos del NetID como NwkID, ya mencionados al describir la De-

vAddr en la Subsección 2.4.2. Estos 7 bits deben ser únicos para redes que funcionan en regiones geográficas cercanas, permitiendo identificar operadores y facilitando el proceso de roaming.

- DevAddr
- DLSettings, un byte disponible para configurar los valores de DR utilizados en las ventanas de downlink.
- RxDelay, un byte que permite configurar el momento en el que se abren las ventanas de downlink.
- CFList, este campo es opcional y está pensado para configurar frecuencias adicionales a ser usadas por el dispositivo.

Las claves de sesión se obtienen aplicando la operación AES *Encrypt* con la clave raíz al AppNonce, NetID y DevNonce (el utilizado en el mensaje de JOIN Request).

En el Apéndice A están detallados los tests de activación que prueban la implementación de estos procesos. Por ejemplo, los tests td_lorawan_act02 y td_lorawan_act03 (Subsección A.1.2 y Subsección A.1.3 respectivamente) intercambian mensajes de JOIN con el dispositivo y prueban que luego la comunicación puede establecerse con los parámetros configurados en el JOIN Accept. La correcta implementación de la configuración de frecuencias usando el campo CFList se prueba en el test td_lorawan_act04 detallado en la Subsección A.1.4.

Una vez presentadas las bases de la tecnología de radio LoRa que dan sustento a LoRaWAN, la arquitectura definida con sus mecanismos de seguridad y el proceso por el cual se activan los dispositivos, lo que resta del capítulo se enfocará en describir los mecanismos definidos para optimizar el funcionamiento de la red una vez iniciada la comunicación.

2.5. Ajuste dinámico y mecanismos de optimización de la red

En esta sección se analizan cómo están implementados los mecanismos para optimizar la red, adaptando de forma dinámica el DR que los dispositivos utilizan e intercambiando mensajes de comandos para realizar ajustes en los parámetros de la comunicación.

El formato definido para los mensajes contempla el intercambio del FHDR que contiene los campos mostrados en la Fig. 2.12.

Este es el significado de cada uno de los campos:

- DevAddr, dirección de red asignada al dispositivo (ver Subsección 2.4.2).
- FCnt, dos bytes con un contador de mensajes.
- FOpts, campo opcional de hasta 15 bytes reservado para comandos de configuración.

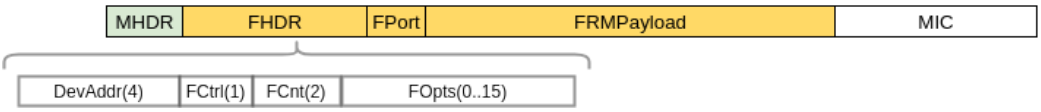

Figura 2.12: Formato del Frame Header (FHDR).

- FCtrl, un byte donde se codifica la siguiente información:
  - activación del mecanismo de Adaptive Data Rate (ADR), explicado más adelante,
  - reconocimiento de mensajes que necesiten confirmación (ACK),
  - largo en bytes del campo FOpts (comandos MAC).

Los mensajes, tanto de Downlink como de Uplink, pueden ser de dos tipos según se indique en el MHDR: con confirmación (*CONFIRMED*) o sin confirmación (*UNCONFIRMED*). Si un mensaje es del tipo *CONFIRMED* quien lo envía queda a la espera de recibir un mensaje de respuesta con el bit de ACK del FCtrl activado, en caso de no recibir esta confirmación el mensaje se vuelve a enviar luego de esperar un tiempo determinado por la región en la que funcione el sistema. En el Apéndice A se detalla el test td_lorawan_fun05 (Subsección A.2.5) que intercambia mensajes con confirmación y el test td_lorawan_fun06 (Subsección A.2.6), donde además se prueba que el dispositivo reenvíe el mismo mensaje en caso de no recibir la confirmación (ACK).

La principal desventaja que presentan los mensajes con confirmación es debida a la limitación adicional que presentan los mensajes de Downlink, que son los que deben contener el ACK reconociendo el mensaje de Uplink recibido. El inconveniente se debe a la tecnología utilizada en los Gateways, que son quienes deben entregar estos mensajes de Downlink, ya que deben dejar de escuchar en todos los canales de recepción durante el tiempo que están transmitiendo un mensaje. Usualmente resulta más conveniente utilizar mensajes de tipo *UNCONFIRMED* y repetir el envío de cada mensaje: todos los nodos tienen configurado un número de repeticiones para los mensajes llamado *NbTrans*.

2.5.1. Protección frente a ataques de replay

En la Subsección 2.4.1 se presentaron los mecanismos de seguridad que protegen al sistema y dificultan ataques que comprometen la integridad o la confidencialidad de los mensajes intercambiados. Una protección adicional, que dificulta los ataques de replay, es la implementación de contadores para los mensajes. Los dispositivos y la red registran la cantidad de mensajes enviados y mantienen un contador de mensajes de Uplink (*FCntUp*) y otro para mensajes de Downlink (*FCntDown*).

Estos dos contadores, junto con la dirección asignada (DevAddr) y claves de sesión, definen el estado de la red al momento de enviar un mensaje y se envían en el campo FCnt del FCtrl en cada mensaje: FCntUp en los mensajes de Uplink

y FCntDown en los de Downlink. Los dispositivos de bajo consumo incrementan sus contadores FCntUp con cada mensaje que envían y mantienen el registro del último FCntDown recibido en un mensaje válido. Análogamente, el Servidor de Red incrementa el contador FCntDown correspondiente al dispositivo destinatario del mensaje y mantiene registro del FCntUp en base al último mensaje válido recibido.

La secuencia de mensajes mostrada en la Fig. 2.13 ejemplifica el proceso de actualización de los contadores. Al recibir un mensaje se comprueba que el FCnt contenido en el mismo sea válido, para lo que tiene que ser mayor al último almacenado [4], y se actualiza este valor. Desde el punto de vista del emisor del mensaje, se envía en el campo FCnt el valor almacenado en el contador correspondiente [5] antes de incrementarlo.

Si un atacante intenta volver a enviar un mensaje antiguo, el FCnt no pasaría la verificación y el mensaje sería descartado por el receptor.

En la Subsección A.2.3 y Subsección A.2.4 del Apéndice A se describen los tests td_lorawan_fun03 y td_lorawan_fun04 que ponen a prueba la implementación de estas comprobaciones por parte del dispositivo, verificando que los números de secuencia sean aumentados correctamente y que se ignoren los mensajes con número de secuencia menor al último recibido.

2.5.2. Configuración Adaptativa del Data Rate

Los dispositivos de bajo consumo pueden elegir libremente el DR al momento de enviar un mensaje. Tal como se explicó en la Sección 2.1, este es un parámetro fundamental para optimizar el funcionamiento de la red y por eso es importante lograr que se utilice el mayor DR posible para reducir el tiempo en el aire de los mensajes, bajando la congestión de la red y aumentando la duración de las baterías.

LoRaWAN define un mecanismo llamado ADR que es activado por el dispositivo y tiene como objetivo optimizar la red mediante la adecuada elección DR y potencias de transmisión. Cuando ADR está activado el Servidor de Red puede especificar los parámetros de transmisión de los mensajes de Uplink enviados por los dispositivos.

Desde el punto de vista de los dispositivos el funcionamiento se basa inicialmente en la utilización del DR más alto posible, siempre que se pueda asegurar que los mensajes están siendo recibidos; en caso de detectar que los mensajes no son escuchados se debe bajar el DR un nivel, hasta llegar al DR más bajo. Cuando el Servidor de Red detecta un mensaje con el bit de ADR activado comienza a monitorear las condiciones de radio del dispositivo y en base a la calidad de la señal recibida (SNR, Signal to Noise Ratio) comienza a enviar comandos de configuración que pueden ajustar los siguientes parámetros en el dispositivo:

[4] Además de ser mayor que el último valor almacenado, no debe superarlo en más de un límite fijado en cada región, llamado *MAX_FCNT_GAP*

[5] Realmente se envían los 16 bits (tamaño del campo FCnt) menos significativos de estos contadores en caso de que se configuren con 32 bits.

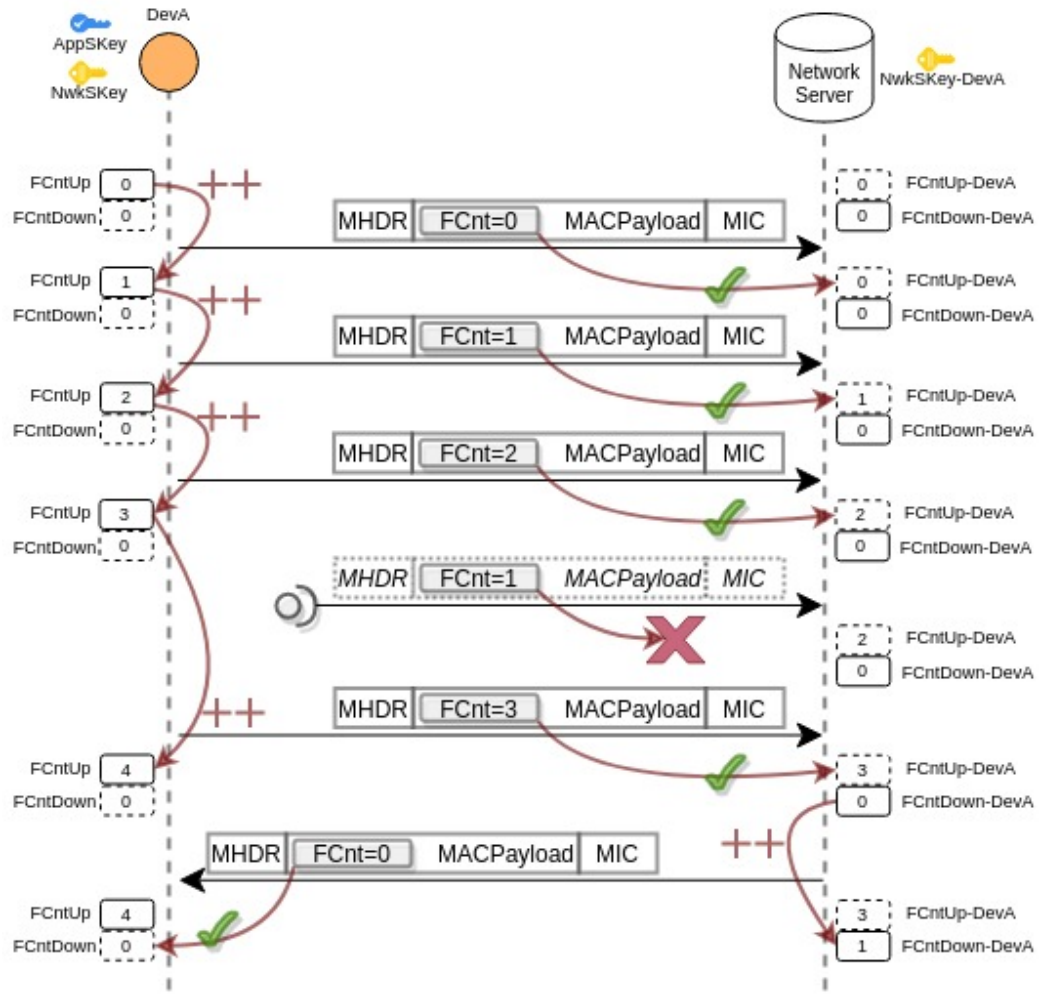

Figura 2.13: Secuencia de mensajes incrementando FCntUp y FCntDown.

- DR recomendado,
- potencia máxima de transmisión,
- canales y frecuencias de comunicación disponibles,
- número de retransmisiones de los mensajes de tipo *UNCONFIRMED* (parámetro *NbTrans*)

La decisión final de si utilizar o no estos parámetros sugeridos es del dispositivo en base a sus capacidades. En la siguiente sección se presentan los comandos de configuración definidos para intercambiar este tipo de información con el dispositivo: comandos MAC. En entornos donde las condiciones de radio sean muy cambiantes, como por ejemplo en situaciones donde el dispositivo sea móvil, el mecanismo de ADR no permite un control óptimo del DR y debe ser desactivado.

Los tests que prueban la implementación del mecanismo ADR quedaron fuera del alcance de este trabajo de tesis, pero tal como se explica en el Capítulo 3 y el Capítulo 4 se espera que la plataforma sea fácilmente ampliable para incorporar nuevos tests.

2.5.3. Comandos de Configuración

Para intercambiar información sobre los parámetros de configuración, como por ejemplo los relativos al mecanismo de ADR presentados en la sección anterior, se definen una serie de comandos de configuración. Estos comandos se intercambian entre la capa MAC (implementación de LoRaWAN) y el Servidor de Red, no interviene en ningún momento el Servidor de Aplicación ni la aplicación implementada en el dispositivo, y por lo tanto son llamados Comandos MAC.

Se definen dos formas diferentes para enviar comandos MAC:

- En el FRMPayload; dedicando un mensaje exclusivamente para los comandos. Se indica que el mensaje contiene comandos MAC poniendo el FPort del mensaje en 0. En este formato de envío los comandos son encriptados utilizando la NwkSKey (ya que su destinatario es el Network Server).
- En el campo FOpts del FHDR; en caso de que el comando quiera enviarse como información adicional a un mensaje de datos. LoRaWAN se refiere a este tipo de comandos como *Piggybacked MAC Commands*, y en la versión 1.0.3 del protocolo estos no están encriptados. El largo máximo del campo FOpts nos limita a una serie de comandos que no exceda los 15 bytes en cada mensaje.

LoRaWAN define que no pueden enviarse comandos MAC de las dos formas en un mismo mensaje, se debe usar el FOpts o el FRMPayload y de haber comandos en ambos campos el mensaje debe ser descartado. En el test td_lorawan_mac02 detallado en la Subsección A.4.2 del Apéndice A se prueba que el dispositivo descarte el mensaje en caso de encontrarse esta condición.

Los comandos MAC consisten en un identificador de comando (*CID, Command Identifier*) y una secuencia de bytes que depende del comando específico que se envía. Debido a que los comandos se procesan en orden, y una vez que se detecta un CID desconocido se asume que no hay más comandos a interpretar, se recomienda formar la lista poniendo los comandos definidos en versiones más recientes de LoRaWAN al final.

Los comandos MAC pueden ser enviados por el Servidor de Red o por el dispositivo de bajo consumo, y para cada CID se define un mensaje de solicitud (*Req*) y otro de respuesta (*Ans*). Basadas en lo especificado en la versión 1.0.3 de LoRaWAN, la Tabla 2.3 resume los comandos MAC enviados por el dispositivo mientras la Tabla 2.4 hace lo mismo con los mensajes enviados por el Servidor de Red.

En las siguientes secciones se explica el funcionamiento y uso de los comandos MAC que fueron implementados en el presente trabajo, señalando la descripción del test correspondiente en el Apéndice.

Comandos enviados por el dispositivo		
CID	Nombre	Descripción
0x02	LinkCheck	Verifica que la conexión con la red funcione correctamente.
0x0D	DeviceTime	Solicita información de fecha y hora a la red LoRaWAN.

Tabla 2.3: Identificadores de los comandos MAC enviados desde los dispositivos hacia la red LoRaWAN.

Comandos enviados por el Network Server		
CID	Nombre	Descripción
0x03	LinkADR	Envia solicitudes de cambio en los parámetros de transmisión y es usado por el mecanismo de ADR.
0x04	DutyCycle	Especifica el máximo Duty Cycle para el dispositivo.
0x05	RxParamSetup	Configura los parámetros de DR de las ventanas de Downlink Rx1 y Rx2 y la frecuencia utilizada para Rx2.
0x06	DevStatus	Solicita información sobre el estado del dispositivo: nivel de las baterías y calidad de la señal recibida.
0x07	NewChannel	Configura canales de comunicación.
0x08	RxTimingSetup	Especifica el delay a utilizarse para abrir las ventanas de recepción Rx1 y Rx2.
0x09	TxParamSetup	Configura el tiempo máximo que puede durar la transmisión de un mensaje (dwell time) así como la potencia máxima de transmisión.
0x0A	DlChannel	Permite modificar la frecuencia utilizada en la ventana de Downlink Rx1, que por defecto utiliza la misma frecuencia que el último mensaje de Uplink.

Tabla 2.4: Identificadores de los comandos MAC enviados desde la red LoRaWAN hacia los dispositivos.

Verificación del estado del dispositivo

Para conocer el estado de un dispositivo el Servidor de Red puede enviar un mensaje de *DevStatusReq*, consistente en un único byte con el CID 0x06. La Fig. 2.14 muestra dos formas diferentes en la que un comando de este tipo puede ser enviado por la red: en la parte superior se ejemplifica con un mensaje enviado exclusivamente con este comando MAC, encriptado, y luego se muestra como se podría incluir este mismo comando en un mensaje de datos (en este caso el comando iría sin encriptar).

El dispositivo debe responder a este mensaje con un comando *DevStatusAns* que, además del CID, consiste de 2 bytes usados para codificar:

- (1) El nivel de batería:
    - 0, para indicar que el dispositivo se conecta a una fuente de alimentación externa.

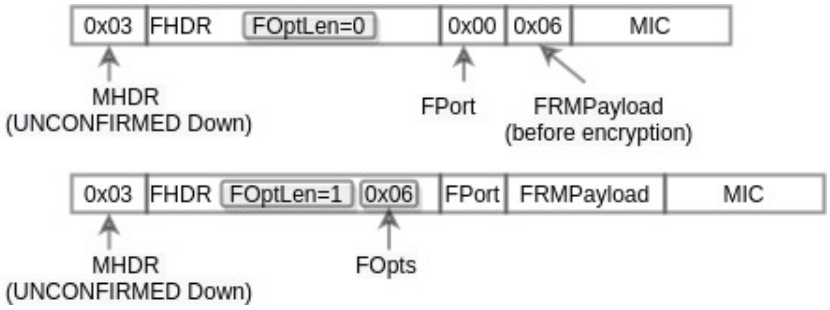

Figura 2.14: Ejemplos de dos formas diferentes de enviar un comando DevStatusReq dentro de un mensaje de Downlink sin confirmación.

- entre 1 y 224, representa el nivel de batería siendo 1 el mínimo.
- 255, indica que no fue posible medir el nivel de batería.

- (2) El margen de demodulación (SNR expresado en dB); 5 bits para representar valores entre -32 y 31.

La Fig. 2.15 muestra dos ejemplos de las respuestas que podría enviar el dispositivo.

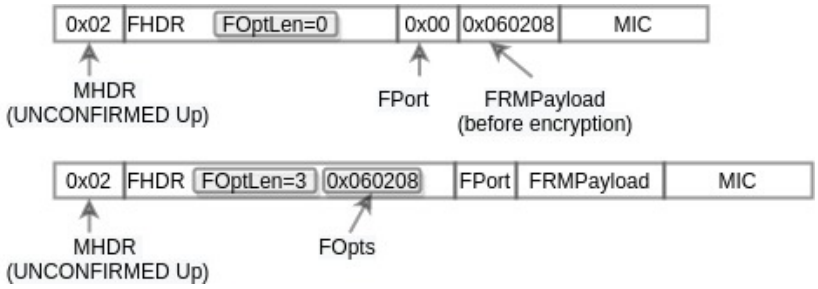

Figura 2.15: Ejemplos de dos formas diferentes de enviar un comando DevStatusAns dentro de un mensaje de Uplink sin confirmación.

El test td_lorawan_mac03 detallado en la Subsección A.4.1 del Apéndice A prueba que el dispositivo responda a este tipo de comandos en los dos formatos posibles: encriptado en el FRMPayload y Piggybacked.

Configuración de frecuencias

Si el Servidor de Red necesita configurar los parámetros de los canales de comunicación utilizados o agregar una nueva frecuencia puede enviar comandos del tipo *NewChannel*. La Fig. 2.16 muestra el formato para la solicitud de un comando del tipo *NewChannelReq*, mientras que la Fig. 2.17 muestra de como una solicitud de este tipo y su respuesta *NewChannelAns* podrían enviarse en el FRMPayload de un mensaje.

En el ejemplo de la Fig. 2.17 se configura el canal 4, usando la frecuencia 868.6 MHZ (Freq=0x0B8984) con un rango de DR aceptados entre DR0 y DR5

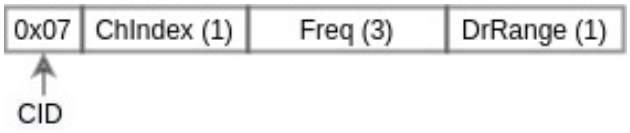

Figura 2.16: Formato de solicitud (Req) de un comando MAC NewChannel (CID=0x07).

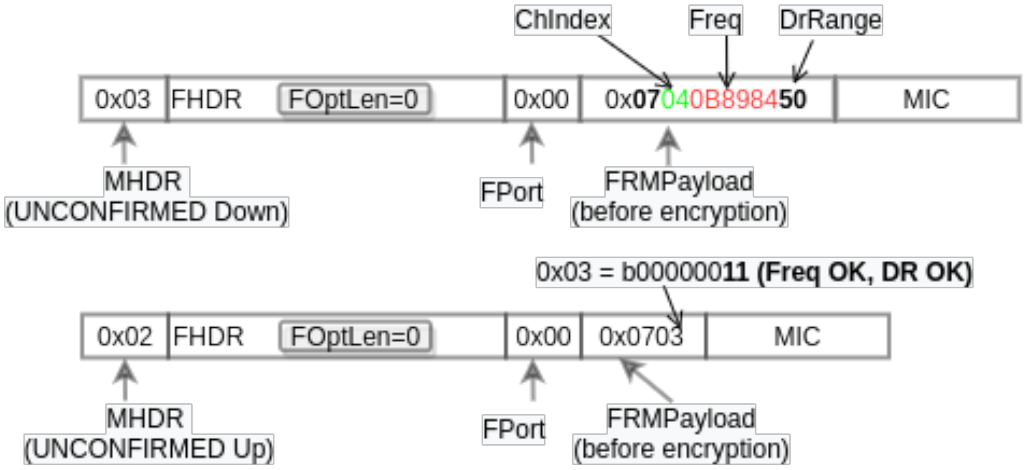

Figura 2.17: Ejemplos de intercambio de *NewChannelReq* y *NewChannelAns*.

(DrRange=0x50). Los tres bytes del campo Freq se interpretan de la siguiente manera: para obtener la frecuencia en Hz se multiplica este valor por 100. En cuanto al DrRange, los 4 bits más significativos representan el máximo DR y los restantes representan el mínimo DR.

La respuesta *NewChannelAns* consta con, además del byte correspondiente al CID, un byte adicional donde los dos bits menos significativos indican si la frecuencia fue aceptada (valor del bit en 1) y si el rango de DR indicado es compatible con el dispositivo (bit en 1).

En el Apéndice A se incluye la descripción de los tests td_lorawan_mac04 (Subsección A.4.4) y td_lorawan_mac05 (Subsección A.4.5) que prueban que la adición de canales funcione correctamente y el test td_lorawan_mac03 (Subsección A.4.3) que intenta eliminar canales por defecto de la región y verifica que el nodo rechace este comando.

Para finalizar este capítulo se incluye una sección en la cual se describen las principales diferencias entre la versión de LoRaWAN usada como base para este trabajo de tesis (v1.0.3) y la especificación de LoRaWAN 1.1. Si bien esta nueva versión de LoRaWAN aún no es muy usada, y hay pocos dispositivos en el mercado que la implementen, es interesante tener en cuenta el camino que la LoRa Alliance está marcando con ella, además de entender el motivo de la introducción de nuevas claves y procesos de activación de dispositivos.

2.6. Mejoras introducidas en LoRaWAN V1.1

La versión 1.1 de LoRaWAN define mejoras en la arquitectura del sistema, cambia el proceso de activación de dispositivos, agrega algunos nuevos comandos MAC y dificulta algunos de los ataques más comunes con mejoras en la seguridad. La especificación de LoRaWAN 1.1 fue publicada por la LoRa Alliance a finales del año 2017, pero aún la gran mayoría de dispositivos implementan las versiones anteriores.

Un factor que explica el tiempo necesario para la adopción de esta nueva versión, además del tiempo relacionado a los procesos de certificación, es que la nueva versión requiere que los dispositivos cuenten con mayores recursos, por ejemplo memoria no volátil, para cumplir con algunas de las mejoras impuestas.

Algunas de los cambios introducidos por LoRaWAN 1.1 incluyen:

- Los contadores de tramas FCntDown y FCntUp son siempre de 32 bits, ya no se permite usar contadores de 16 bits, y se envía siempre los bits menos significativos en el campo FCnt de los mensajes. Además, estos no pueden ser reutilizados en una misma sesión por lo que si un dispositivo se reinicia debe recordar el valor del contador e iniciar una nueva sesión si quiere volver a ponerlos a cero.
- Desde la red se mantienen dos contadores diferentes para las tramas de Downlink, uno a nivel de red (*NFCntDown*) o otro a nivel de aplicación (*AFCntDown*).
- Se agregan comandos MAC adicionales.
- Se dificultan algunos tipos de ataque relacionados con el reconocimiento de los mensajes (ACK), ya que en LoRaWAN 1.1 se tiene en cuenta el FCnt del mensaje que se está reconociendo para calcular el MIC. En las versiones 1.0.x cuando se recibe un mensaje con ACK no se puede determinar con seguridad cuál es el mensaje del que se acusa recibo.
- Los comandos MAC deben ir siempre encryptados, incluso los que son enviados el el campo FOpts del FHDR.
- El DevNonce usado en el proceso de JOIN en las versiones 1.0.x es un valor generado aleatoriamente, y el servidor de red guarda un listado de los últimos utilizados. En LoRaWAN 1.1 este parámetro del mensaje de JOIN Request es un contador que se incrementa en cada proceso de JOIN. Además, se le cambia el nombre a JoinNonce.

2.6.1. Nueva arquitectura con Join Server

La principal diferencia introducida en LoRaWAN 1.1 se encuentra en el cambio de arquitectura, con la especificación de un Join Server encargado de almacenar las claves raíz y de manejar el proceso de Join. De esta forma se facilita el despliegue de los dispositivos en diferentes redes, independizando a las aplicaciones del proveedor

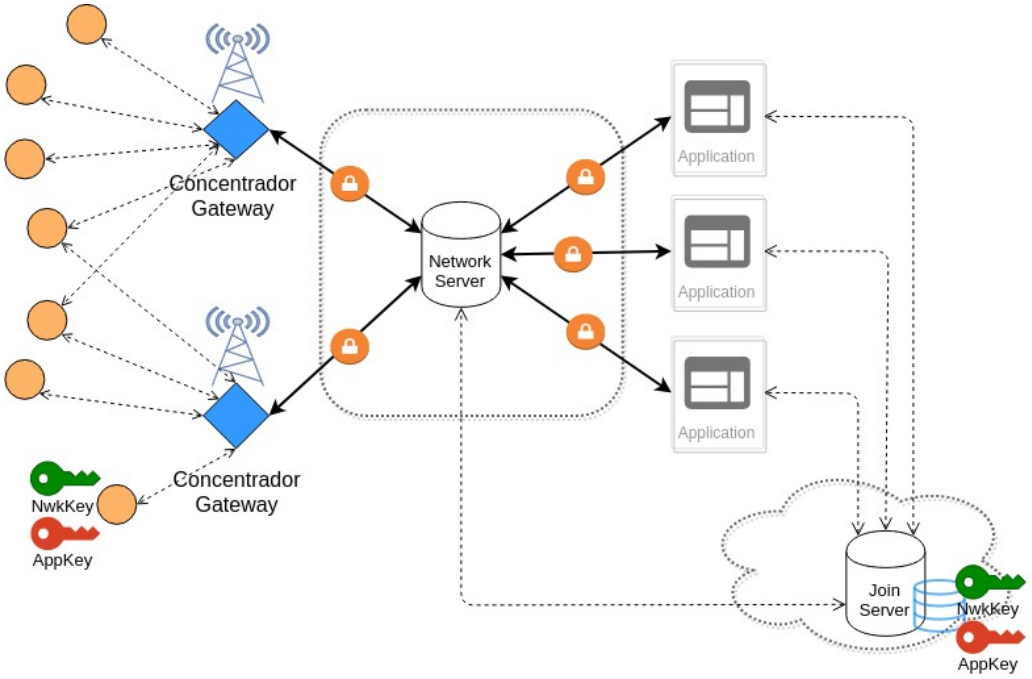

Figura 2.18: Claves raíz en la arquitectura de una red LoRaWAN v1.1

de red. La Fig. 2.18 muestra cómo ahora el Servidor de Red y los Servidores de Aplicación ya no guardan ninguna clave raíz y es un servidor independiente a la red quien se encarga de esto. Ahora cuando el dispositivo inicia el proceso de activación se comunica con el Join Server (de ahí el cambio de AppEUI a JoinEUI en el campo del mensaje JOIN Request, explicado en la Sección 2.4.3) y es este servidor quien se encarga de comunicar las claves de sesión generadas al Servidor de Red y a la aplicación correspondiente.

2.6.2. Manejo de claves raíz

Una vulnerabilidad en la seguridad de LoRaWAN se encuentra en el manejo de las claves raíz, que deben ser grabadas en elementos seguros en cada dispositivo al momento de fabricarla y luego se expone todo el sistema al tener que comunicar las claves de la red en las que va a funcionar (con la dificultad añadida de gestionar los potenciales cambios de un proveedor de red por otro). Con la introducción del Join Server se habilita una arquitectura en la que por ejemplo el fabricante puede gestionar las claves y comunicarlas a la red LoRaWAN en que se desee hacer funcionar el dispositivo.

LoRaWAN 1.1 agrega una nueva clave raíz para cada dispositivo llamada *Network Key (NwkKey)* adicionalmente a la ya conocida *AppKey*, por lo que en esta versión la información mínima necesaria en un dispositivo para iniciar sesión usando OTAA es:

- DevEUI

- JoinEUI
- AppKey
- NwkKey

Esta nueva clave raíz colabora junto a la introducción del Join Server en la separación de las responsabilidades y privilegios que se le asigna al proveedor de red. En LoRaWAN 1.1 la clave de sesión de aplicación *AppSKey* se sigue derivando de la *AppKey*, pero para la generación de las claves de sesión utilizadas por la red para control de integridad y encriptación de comandos se utiliza la *NwkKey*. De esta forma se podría dar la clave raíz *NwkKey* a un proveedor de red para que gestione sus claves de sesión de red sin comprometer la confidencialidad a nivel de aplicación.

El Join Server deriva dos claves adicionales que son utilizadas para el proceso de JOIN: *JSIntKey* y *JSEncKey*. Estas claves se conservan iguales luego de cada cambio de sesión y se utilizan para encriptar los mensajes de JOIN Accept (que hasta LoRaWAN 1.0.3 son encriptados usando la *AppKey*) y para calcular el MIC en el proceso de JOIN.

2.6.3. Flexibilidad para el roaming y claves de sesión

Debido a que los Gateways LoRa no pueden determinar el dispositivo que le está enviando datos, todos los mensajes que estos reciben son enviados al Servidor de Red aunque pertenezcan a otras redes. Esto presenta una oportunidad para que las diferentes redes se conecten para intercambiar los mensajes recibidos, por lo que definir elementos en la arquitectura que contemplen la posibilidad de roaming resulta interesante.

En todas las versiones de LoRaWAN la dirección asignada por la red al dispositivo (*DevAddr*) contiene un campo que identifica a la red a la que pertenece (derivado del *NetID* de la red). Esto permite que un mensaje que es recibido por otro Servidor de Red, porque por ejemplo el dispositivo se haya movido a una región cubierta solo por Gateways de otra red, pueda ser dirigido a la red a la que pertenece en un proceso conocido como *passive roaming*.

LoRaWAN 1.1 flexibiliza las forma de implementar el roaming entre redes introduciendo el que se conoce como *handover roaming*. En este proceso de roaming el dispositivo sabe que está en situación de roaming y puede intercambiar comandos de configuración con el Servidor de Red de su zona de cobertura de red; la especificación hace referencia a este servidor de red como *Serving Network Server*, en contraste con el *Home Network Server* donde el dispositivo está registrado.

Se definen las siguientes claves de sesión:

- A partir de la AppKey:
  - AppSKey
- A partir de la NwkKey:

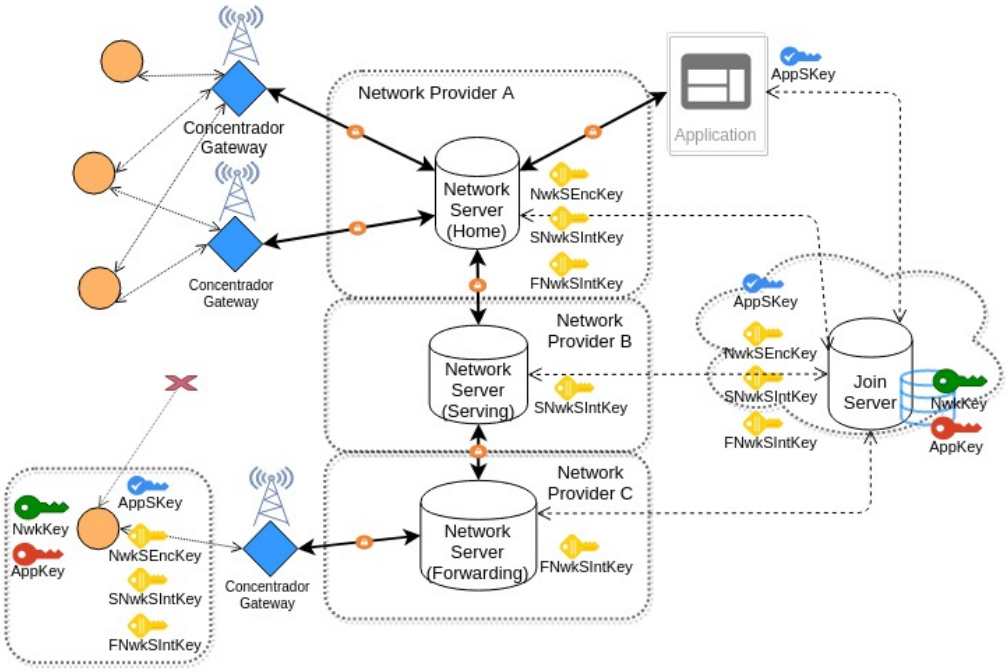

Figura 2.19: Arquitectura

- NwkSEncKey
- FNwkSEncKey
- SNwkSEncKey

La Fig. 2.19 muestra un escenario de ejemplo con un dispositivo activado y enviando datos a través de un Servidor de Red de otro proveedor. Las claves de sesión de red permiten que las otras redes hagan comprobación de parte del MIC sin otorgar más privilegios de los necesarios.

En cuanto a compatibilidad, un dispositivo que implementa la versión 1.1 será compatible con Servidores de Red 1.1 o 1.0.x, mientras que un dispositivo que implementa las versiones 1.0.x necesita un Servidor de Red compatible con 1.0.x. El proceso de JOIN define ahora señalización para indicar a un dispositivo compatible con v1.1 que debe adaptarse a la v1.0.x y usar la *NwkKey* para generar las claves de sesión. La tendencia es que los servidores de red comienzan a soportar dispositivos LoRaWAN de ambas versiones.

2.7. Resumen y Conclusión del Capítulo

En este Capítulo se presentaron las bases del estándar definido por LoRaWAN para lograr un sistema de comunicación basado en una capa física LoRa. Una comprensión de la arquitectura, los mecanismos de seguridad implementados así como de los procesos de activación de los dispositivos permite comprender qué tipo

de pruebas deben implementarse para comprobar que un dispositivo implementa correctamente el estándar.

Se explicó el rol de la LoRa Alliance, haciendo un breve resumen de los documentos y procesos que esta define para ayudar a que LoRaWAN sea implementado con éxito y garantizando la interoperabilidad de dispositivos. Mostrando la forma de trabajo con distintas versiones del estándar se dejó claro cuál es la versión de referencia utilizada en este trabajo de tesis y se incluyó finalmente un breve resumen de las principales diferencias introducidas en LoRaWAN 1.1. Se espera que esto sirva como referencia para eventuales trabajos futuros que deseen ampliar el alcance de esta plataforma de pruebas.

Una vez estando claros los objetivos y con una comprensión del funcionamiento de la tecnología de comunicación que se pondrá a prueba, en los próximos dos capítulos se presentará el diseño de la plataforma de tests de conformidad para LoRaWAN y las tecnologías y criterios para su implementación.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Diseño de la Plataforma de Tests

Los tests de conformidad son una de las maneras indicadas por la ETSI [13] para probar que la implementación de un protocolo de comunicación cumple con un determinado estándar. En este tipo de tests una implementación particular de un protocolo de comunicación (e.g. LoRaWAN) se prueba frente a una plataforma de tests que establece una secuencia de pasos a ejecutar.

En este capítulo se describe el diseño de una plataforma de tests de conformidad para LoRaWAN, mostrando la arquitectura general y explicando el rol de los distintos componentes del sistema. Se muestra la definición y diseño de la estructura que siguen los diferentes tests, así como el protocolo usado para lo mensajes que son intercambiados entre la plataforma de pruebas y el dispositivo que está siendo probado.

3.1. Tests de conformidad

Para verificar que se está compliendo con los requerimientos del estándar, los tests de conformidad realizan pruebas sobre el contenido y formato de los mensajes así como sobre la secuencia en la que estos son intercambiados [13].

Al realizar los tests se determina si un dispositivo cumple con la especificación del estándar cuando interactúa con una plataforma de tests de conformidad, como se muestra en la Fig. 3.1.

Figura 3.1: Tests de conformidad sobre un dispositivo (DUT, Device Under Test).

De acuerdo a la arquitectura definida por LoRaWAN y explicada en la Sección 2.4 del Capítulo 2 la plataforma de tests de conformidad deberá comportarse asumiendo los roles de los servidores de red y aplicación.

El conjunto de tests implementados por la plataforma debe probar que un dispositivo utiliza un formato correcto para los mensajes de datos y de configuración siguiendo una versión particular de la especificación de LoRaWAN. Adoptando los mismos lineamientos de la LoRa Alliance, se dividen los tests a realizar en los siguientes grupos:

- ACT, tests de activación. Se comprueba el funcionamiento del inicio de sesión usando los mecanismos ABP y OTAA.
- FUN, funcionalidades básicas y temporización. Estos tests verifican que se pueden enviar mensajes de forma correcta en ambas ventanas de downlink, el manejo de los números de secuencia y las retransmisiones de mensajes.
- SEC, seguridad, encriptación y verificación de integridad de los mensajes. Se envían mensajes destinados a verificar que la encriptación del payload y el cálculo del MIC se hace de forma correcta.
- MAC, implementación de los comandos MAC. Se envían comandos MAC y se verifica que el Device Under Test (DUT) responde de acuerdo a la especificación.

3.2. Proceso de certificación de la LoRa Alliance

La LoRa Alliance define un proceso de certificación de productos, luego del cual permite que estos sean promocionados e identificarlos como "LoRaWAN Certified". Este proceso está enfocado en asegurar que un producto cumple con la especificación de LoRaWAN, y que por lo tanto cumplirá con las expectativas de calidad al conectarse a una red LoRaWAN, evitando de esta forma costos de soporte y la aparición de problemas luego de que el producto fue desplegado.

Para poder aplicar a este tipo de certificación es necesario ser miembro de la LoRa Alliance [1] y realizar el proceso de tests en una Authorized Test House [2].

El proceso de certificación divide las pruebas de acuerdo a la región que se quiere certificar, ya que el dispositivo debe cumplir con un documento de parámetros específico para cada región, y a la versión de la especificación de LoRaWAN. Actualmente la LoRa Alliance cuenta con programas de certificación para las siguientes regiones [3]:

- Europa: EU863-870 MHz.
- America del Norte: US902-928 MHz
- Asia: AS923 MHz.
- Corea: KR920-923 MHz.

[1] `https://lora-alliance.org/lorawan-certification`
[2] `https://lora-alliance.org/certification-test-houses`
[3] `https://lora-alliance.org/lorawan-certification`

- India: IN865-867 MHz.

El trabajo de esta tesis se centró en la región European Union 863-870 MHz ISM Band con la versión 1.0.3 de LoRaWAN. Todos los test implementados se limitan a probar que un DUT podría pasar las pruebas de certificación en esta región y para esta versión del protocolo. Tests adicionales pueden ser implementados para cubrir otras regiones, en particular la región AU915-928 utilizada en Uruguay, o para agregar pruebas de funcionalidades o requerimientos adicionales de otras versiones del protocolo, pero eso ha quedado fuera del alcance de este trabajo de tesis.

Las pruebas que se realizan durante el proceso de certificación no incluyen tests para verificar que el dispositivo cumple con las regulaciones de radio, ni se realiza ninguna prueba de *performance*. Todos los tests se centran en certificar que el nodo cumple con los requerimientos mínimos definidos por la especificación de LoRaWAN, en cuanto al formato de mensajes, tiempos de las ventanas de recepción, mecanismos de encriptación y uso de frecuencias. El certificado otorgado luego de pasar las pruebas indica específicamente cuáles de las funcionalidades opcionales del protocolo son cumplidas según el estándar en caso de que estas pruebas adicionales hayan sido aprobadas.

La Fig. 3.2, extraída de la web de la LoRa Alliance, resume el proceso de certificación. Luego de que un miembro de la LoRa Alliance contacta a una casa autorizada para certificaciones (ATH, Authorised Test House), completa los formularios solicitados y prepara el producto para la certificación, un conjunto de tests son realizados y si alguno de los requerimientos no son coumplidos por la implementación de LoRaWAN se deben hacer ajustes y volver a realizar las pruebas.

Para evitar tener que hacer ajustes al momento de someterse a las pruebas de certificación, los desarrolladores deberían implementar sus propios tests. Una forma de facilitar el proceso de desarrollo y ayudar a minimizar el tiempo necesario para lanzar un producto es proporcionar herramientas que permitan probar una implementación antes de someterla al proceso de certificación de LoRaWAN.

3.3. Arquitectura de la Plataforma de Tests

Un servidor de aplicación de test, Test Application Server (TAS), será el componente central de la plataforma de tests, siendo quien maneja todas las interacciones con los dispositivos, DUT, que se someten a pruebas. Para probar una implementación de LoRaWAN (IUT, Implementation Under Test) se inicia una Sesión de Test que está manejada por un *Test Manager*. Este Test Manager tiene la lista de tests a ejecutarse y es quien maneja todas las interacciones con el DUT.

Se puede apreciar en la Fig. 3.3 que cada test está organizado como una secuencia de pasos (Steps). En cada paso se prueba una funcionalidad concreta verificando que el DUT responde tal como está determinado por la especificación de LoRaWAN.

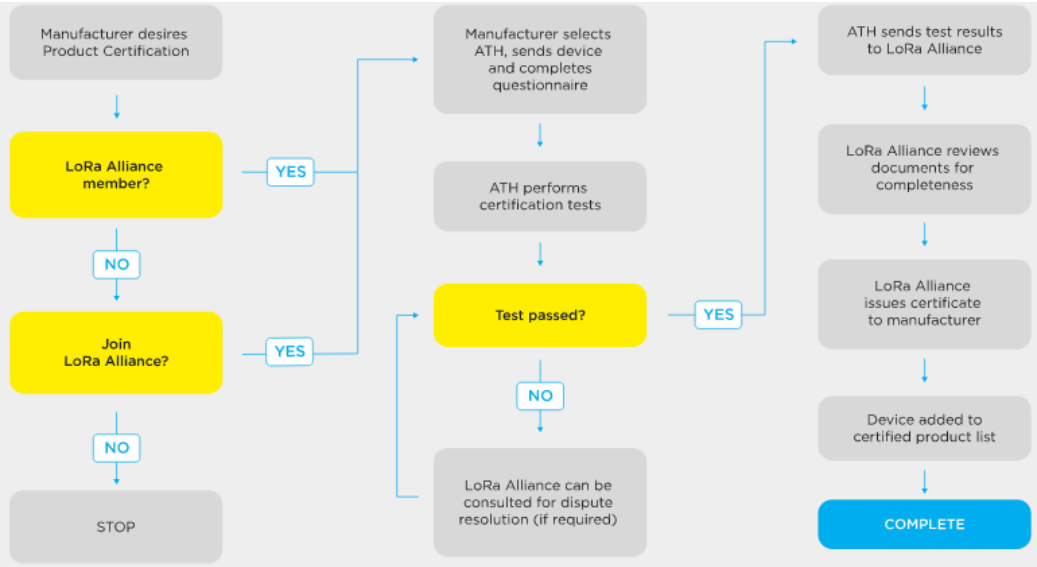

Figura 3.2: Proceso de Certificación de la LoRa Alliance.

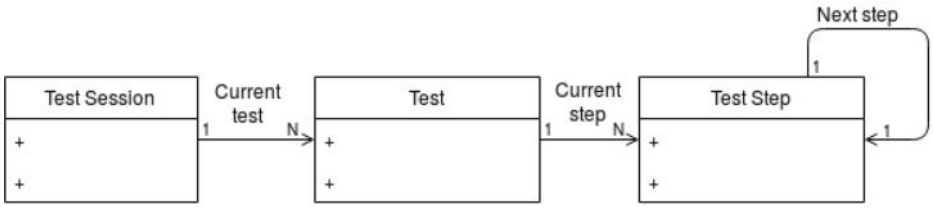

Figura 3.3: Diagrama de tests de una sesión y los pasos que lo componen.

El DUT debe implementar sobre la capa LoRaWAN a testear (IUT) un protocolo de aplicación de tests que interactúa con el Test Manager. En cada paso del test se realiza un intercambio de mensajes entre el DUT y el Test Manager, donde este último componente verifica que el tipo de mensaje recibido y su formato sea el esperado de acuerdo a la especificación de LoRaWAN. Si en un paso del test se detecta algún tipo de error el resultado del test es FAIL y se continúa con el siguiente test.

3.3.1. Componentes del sistema

La Fig. 3.4 muestra los componentes de la plataforma y los que debe proporcionar el usuario para probar una implementación de LoRaWAN.

El usuario de la plataforma de tests debe proporcionar, además del DUT con la IUT, un Gateway LoRa con una aplicación de Packet Forwarder configurado para reenviar los mensajes del nodo hacia el TAS. Idealmente, el DUT debe ubicarse en un contenedor electromagnéticamente aislado de otras comunicaciones inalámbricas tal como se realizan las pruebas de conformidad de LoRaWAN.

Entre los principales componentes de la plataforma, además del ya mencionado

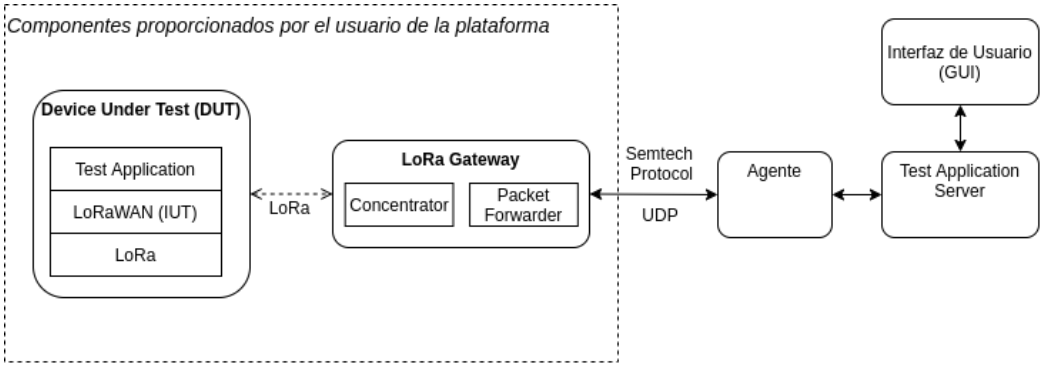

Figura 3.4: Arquitectura básica de la plataforma de tests.

TAS, se encuentra una Interfaz de Usuario que permite visualizar el progreso de las pruebas y el reporte de resultados así como enviar comandos para configurar la sesión de test. El Agente de usuario funciona como un conector entre el LoRa Gateway (proporcionado por el usuario) y el TAS. La comunicación entre estos componentes usa el protocolo definido por Semtech para el packet forwarder a través de una conexión UDP en la red local.

User Agent

El Agente de usuario canaliza toda la comunicación entre el LoRa Gateway que envía los mensajes de tests y el TAS. Incluir al Agente en el diseño, además de adaptar el protocolo utilizado por el Packet Forwarder, permite que la plataforma pueda ser integrada en otros sistemas de tests en un escenario en el que el TAS y la interfaz de usuario se ejecuten en un sistema remoto (e.g. Cloud Server).

La Fig. 3.5 muestra el escenario donde toda la plataforma se ejecutaría en local, en un PC del usuario, mientras que la Fig. 3.6 describe el escenario en el que el TAS estaría integrado en una plataforma de tests externa o remota a la que el Agente se conectaría de forma segura enviando los mensajes recibidos desde el LoRa Gateway.

El LoRa Gateway ejecutará una aplicación de packet forwarder diseñada para enviar a un servidor de red LoRaWAN todos los mensajes recibidos utilizando el protocolo definido por Semtech (Semtech Protocol en Fig. 3.4) [4].

Este protocolo se encuentra disponible en una gran cantidad de gateways y por ello es utilizado para la plataforma de pruebas de conformidad de LoRaWAN. Si bien tiene limitaciones desde el punto de vista de la seguridad, al utilizar un agente que encapsula y reenvía los mensajes recibidos desde el Packet Forwarder se asegura que estos son encriptados y se realiza una autenticación de quién envía los mensajes al TAS. Este punto es particularmente importante en el escenario de un TAS remoto.

[4] `https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT`

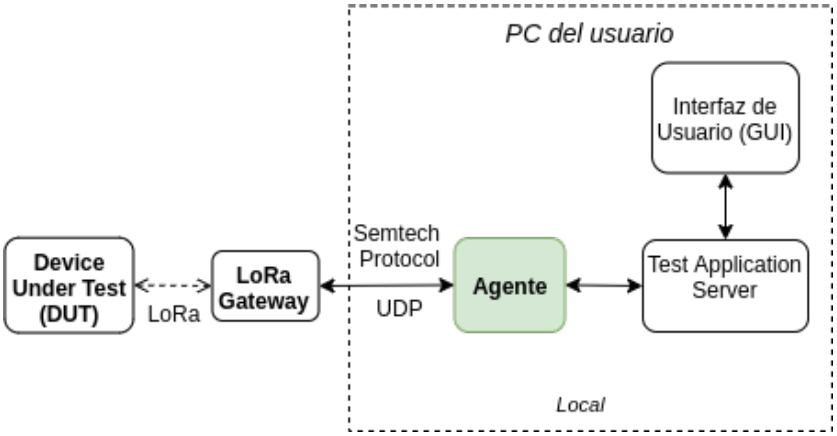

Figura 3.5: TAS y Agente de usuario ejecutándose en un PC del usuario.

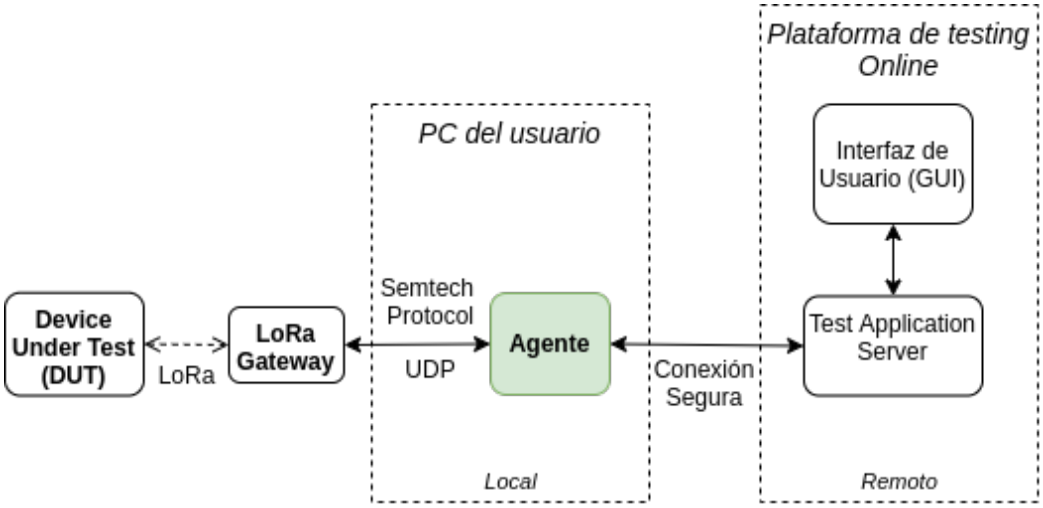

Figura 3.6: Agente de usuario conectándose con un TAS remoto.

Interfaz de usuario

Una interfaz de usuario permite iniciar una sesión de test, configurar los parámetros del DUT como claves de encriptación y EUIs, así como seleccionar los tests que se desean ejecutar. A través de la interfaz de usuario se muestran los reportes de resultados de los tests con detalles de los pasos que han fallado en caso de que corresponda.

El presente trabajo de tesis no tiene entre sus objetivos prioritarios el desarrollar una interfaz de usuario gráfica avanzada, sino que simplemente se proporcionan formas de interactuar mediante línea de comandos, acceso a los logs y una interfaz web básica opcional.

La Fig. 3.7 muestra esquemáticamente los componentes de la interfaz de usuario, que proporcionan una interfaz por línea de comandos (*CLI*) y un visualizador (*Notification Displayer*) que interactúan con el TAS. El diagrama también ejempli-

fica cómo estos servicios pueden ser utilizados para luego implementar una interfaz gráfica sobre ellos.

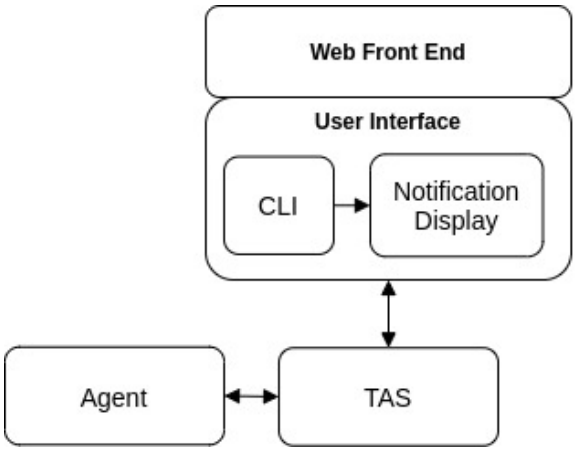

Figura 3.7: Diseño de la interfaz de usuario.

Diseño del Test Application Server

El TAS mantiene la sesión y controla toda la interacción con el DUT. Cada sesión de test en el TAS es controlada por un coordinador de sesión que almacena las credenciales y la configuración del DUT, conoce la secuencia de tests a ejecutarse y se encarga de procesar la información entregada por cada test con los resultados antes de ser presentada a la interfaz de usuario. La Fig. 3.8 muestra un esquema de los componentes básicos del TAS.

El coordinador de la sesión (*TestSessionCoordinator*) tiene un registro del DUT (*End Device Register*) con las credenciales e información de sesión y contadores de trama (Frame Counters, *fcnt_up* y *fcnt_down*) de LoRaWAN. El registro del DUT es capaz de generar mensajes LoRaWAN de tipo *JOIN Accept* en respuesta de un JOIN Request recibido desde el DUT. Además, al conocer las claves de sesión, es capaz de preparar el PHY Payload de LoRaWAN con todos los encabezados y formato correcto a partir del payload que se desea enviar como mensaje de downlink hacia el DUT.

Los tests están compuestos por un Test Application Manager, componente que conoce la secuencia de pasos a ejecutarse en el test. Cada uno de los pasos del test (Steps) es capaz de verificar que algún aspecto específico de LoRaWAN se cumple correctamente. Para facilitar la reutilización de componentes y poder hacer un desarrollo modular de los tests, la comprobación en cada Step se divide en un chequeo básico que realiza comprobaciones simples al inicio del test y un step handler que se encarga del resto de las verificaciones que deban realizarse en ese paso.

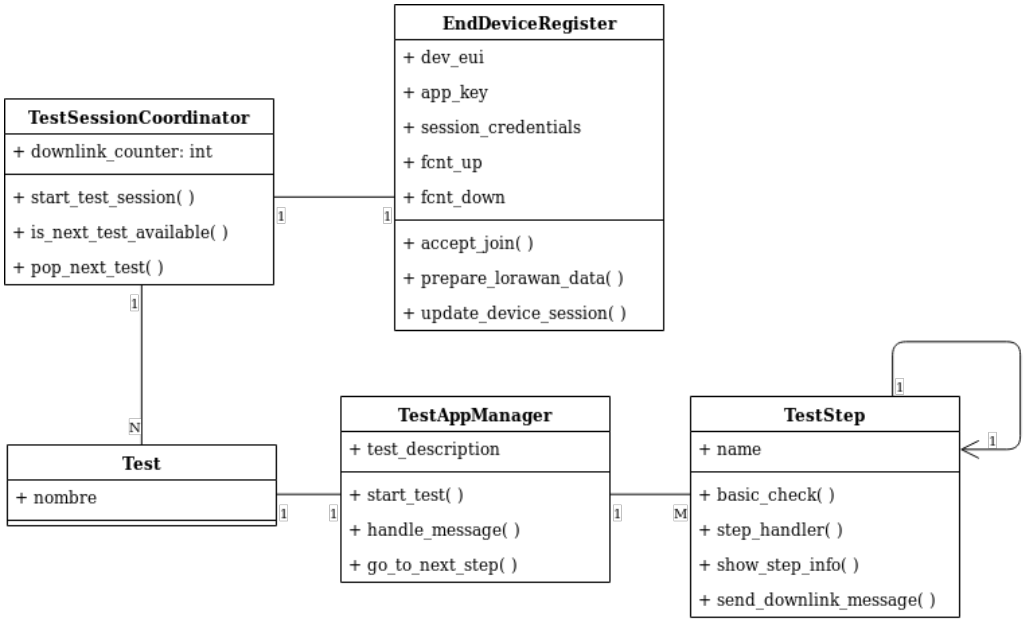

Figura 3.8: Diseño del TAS.

3.4. Test Application Protocol

Esta sección describe los diferentes mensajes intercambiados entre los DUT y el TAS.

Los DUT son nodos de bajo consumo que se comunican utilizando una implementación del protocolo LoRaWAN en su capa MAC, que será la implementación a testear (IUT). Utilizando los servicios de la capa MAC de LoRaWAN el DUT deberá implementar una capa de aplicación que sea capaz que comunicarse con el TAS utilizando un protocolo de test que define diferentes tipos de mensajes y especifica el comportamiento que debe seguir un nodo como respuesta a los estímulos que recibe desde la plataforma de testing. La Fig. 3.9 muestra esquemáticamente la interacción entre el DUT y el TAS, con un intercambio de mensaje que es controlado e iniciado por el TAS. El puerto 224 es utilizado para el intercambio de los mensajes de test, según está definido por la especificación de LoRaWAN.

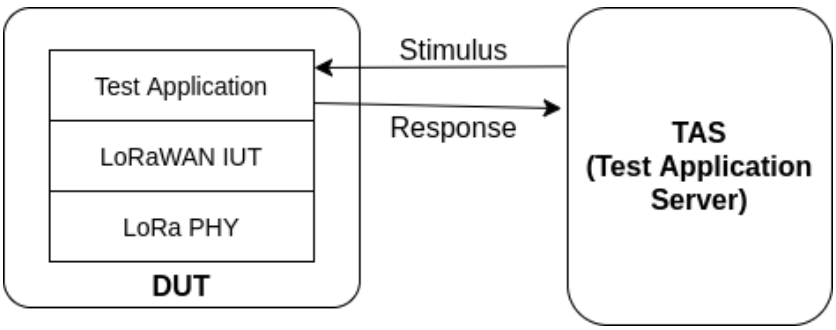

Figura 3.9: Stack de protocolos implementados en el DUT.

Cuando un DUT se comunica con el TAS, iniciando una sesión tal como lo haría ante cualquier servidor de red LoRaWAN, recibe como respuesta un mensaje de activación que le indica que debe pasarse a modo de test. A partir del momento en que el DUT pasa a estar en modo test debe enviar un mensaje periódico de Test Activation OK (TAOK), con un período de entre 5 y 10 segundos. Este mensaje de TAOK tiene una longitud de dos bytes y consiste en un contador que es incrementado cada vez que el DUT recibe un mensaje de test procedente del TAS.

Durante las pruebas de conformidad, mientras se encuentra en modo test, el DUT debe suspender el envío de mensajes de datos y no se espera que cumpla con las restricciones impuestas por la región de radio utilizada. Si el DUT y el LoRa Gateway están aislados del entorno este incumplimiento no afectará a ningún otro sistema.

El DUT debe implementar el mecanismo ABP. Además, el protocolo de test contempla el envío de mensajes de *Join Trigger* que indican al DUT que debe solicitar una nueva sesión en la red mediante un *JOIN Request* al que el TAS responde con un mensaje LoRaWAN de *JOIN Accept*. El DUT pasa a estado de funcionamiento normal luego de recibir *JOIN Accept* (desactiva el modo de test) y envía un mensaje de datos inmediatamente después de iniciar sesión, en general cada vez que el dispositivo inicia una nueva sesión se deberá activar nuevamente el modo test. Este tipo de mensajes se utilizarán en dispositivos que implementan el mecanismo de OTAA.

La Tabla 3.1 resume los diferentes mensajes que son intercambiados entre el TAS y el DUT utilizando el puerto (FPort) 224.

Tabla 3.1: Mensajes definidos en el protocolo de tests a nivel de aplicación. El Test ID corresponde al primer byte del FRMPayload.

Test ID	Nombre	Descripción
0x00	Desactivación del Modo Test	Indica al DUT que debe desactivar el modo test.
0x01	Activación del Modo Test	Activa el modo de test en el DUT enviando un mensaje con FRMPayload=0x01010101. Luego de recibir este mensaje el DUT comienza a enviar mensajes de TAOK con el contador de downllink.
0x02	Confirmed Uplink	Indica a la aplicación de test en el DUT que a partir de este momento debe utilizar mensajes de tipo CONFIRMED uplink en todas las transmisiones.
0x03	Unconfirmed Uplink	Indica a la aplicación de test en el DUT que a partir de este momento debe utilizar mensajes de tipo UNCONFIRMED uplink en todas las transmisiones.

Test ID	Nombre	Descripción
0x04	Ping Pong	Utilizado para comprobar los mecanismos de encriptación, se envía un mensaje de largo y contenido aleatorio que comienza siempre con el byte 0x04. El DUT debe responder con un mensaje que también comienza con el byte 0x04 y para los siguientes se incrementa en 1 (módulo 256) el valor del byte recibido. Por ejemplo, ▪ Ping request: 0x04**ca**32f5a5b7f1187583d3 ▪ Pong response: 0x04**cb**33f6a6b8f2197684d4
0x06	Session Update	Solicita al DUT que inicie una nueva sesión LoRaWAN mediante el envío de un mensaje de JOIN REQUEST.

3.5. Diseño de los tests de conformidad

La presente sección inicia con un listado de los tests implementados, que son un subconjunto de los definidos por la LoRa Alliance y realizados en el proceso de certificación de LoRaWAN. Luego se pasa a explicar cómo es el diseño de cada test usando como ejemplo uno en particular y en el Apéndice A se encuentran detallados todos pasos que se ejecutan en todos los tests implementados.

En la Tabla 3.2 se describen brevemente todos los tests implementados indicando el grupo al que pertenecen: ACT, FUN, SEC o MAC.

Tabla 3.2: Lista de tests implementados.

Número (Grupo)	Test ID	Resumen del Test
1 (ACT)	td_lorawan_act_01	Activa el Modo Test y prueba que se puede establecer comunicación utilizando el mecanismo de Activation By Personalization (ABP).
2 (ACT)	td_lorawan_act_02	Prueba la activación de tipo Over the Air Activation (OTAA) cambiando los parámetros de Data Rate (DR) de las ventanas de downlink y verificando que se puede activar el dispositivo en ambas (RX1 y RX2).

Número (Grupo)	Test ID	Resumen del Test
3 (ACT)	td_lorawan_act_03	Prueba la activación OTAA en ambas ventanas de downlink luego de cambiar el delay de las mismas.
4 (ACT)	td_lorawan_act_04	Prueba la activación de tipo OTAA agregando 5 nuevos canales de comunicación.
5 (ACT)	td_lorawan_act_05	Reinicia los parámetros por defecto definidos por LoRaWAN en el DUT mediante el uso de un mensaje JOIN Accept (activación OTAA).
6 (FUN)	td_lorawan_fun_01	Prueba básica de comunicación mediante el intercambio de mensajes de tipo PING PONG.
7 (FUN)	td_lorawan_fun_02	Prueba la tolerancia del DUT ante errores de tiempo en el envío de mensajes de downlink en las ventanas RX1 y RX2. Verifica que se acepten mensajes con la tolerancia especificada por LoRaWAN de +/- 20 microsegundos.
8 (FUN)	td_lorawan_fun_03	Verifica que los números de secuencia de las tramas sucesivas sean incrementados correctamente.
9 (FUN)	td_lorawan_fun_04	Prueba que el DUT rechace mensajes que tienen un número de secuencia menor al de uno recibido previamente.
10 (FUN)	td_lorawan_fun_05	Verifica el manejo de los reconocimientos (ACK) para los mensajes de tipo CONFIRMED.
11 (FUN)	td_lorawan_fun_06	Prueba la retransmisión de los mensajes de uplink de tipo CONFIRMED en caso de que los mismos no sean reconocidos.
12 (SEC)	td_lorawan_sec_01	Se intercambian mensajes de tipo PING PONG de distintos tamaños para verificar que el MIC es calculado correctamente y los mecanismos de encriptación están bien implementados.
13 (SEC)	td_lorawan_sec_02	Se envía al DUT un mensaje con errores en el MIC y se verifica que el mismo sea ignorado.
14 (MAC)	td_lorawan_mac_01	Prueba el envío de comandos MAC del tipo DevStatus en el payload (FRMPayload) y en el campo FOpts del encabezado de trama.

Número (Grupo)	Test ID	Resumen del Test
15 (MAC)	td_lorawan_mac_02	Verifica que un mensaje enviado al DUT conteniendo mensajes MAC simultáneamente en el payload y en el campo FOpts del encabezado de trama es ignorado.
16 (MAC)	td_lorawan_mac_03	Intenta modificar en el DUT los canales por defecto de LoRaWAN para la región y verifica que el DUT no responde a este pedido.
17 (MAC)	td_lorawan_mac_04	Utiliza un comando MAC (NewChannelReq) para agregar varios canales de comunicación en el DUT.
18 (MAC)	td_lorawan_mac_05	Utiliza un comando MAC (NewChannelReq) agregar un único canal de comunicación.
19 (-)	td_lorawan_deactivate	Envía el comando de desactivación del Modo Test para dejar al dispositivo funcionando normalmente.

El diseño de los tests de conformidad, siguiendo lo explicado en la Sección 3.3.1, se basa en un *Test Manager* que tiene la secuencia de pasos a ejecutarse (i.e. *Steps*). En cada uno de estos pasos se espera la recepción de un mensaje desde el DUT, se realiza una verificación de que el formato del mensaje es correcto y eventualmente se envía una respuesta en un mensaje de *Downlink* al DUT. Esta respuesta puede ser un estímulo que indique al DUT que debe cambiar de modo de funcionamiento, iniciar una nueva sesión, o realizar alguna acción según el protocolo de aplicación de tests explicado en la Sección 3.4.

Se selecciona el test *td_lorawan_act_02* para analizar detalladamente a modo de ejemplo. Este test de conformidad, que correspondiente al grupo de tests de activación (ACT), prueba que la activación de tipo OTAA está correctamente implementada en la IUT.

La Fig. 4.13 muestra un diagrama de secuencia con los pasos que componen el test. Si alguno de los pasos detecta un error, el resultado del test será FAIL y se pasará al siguiente test, mostrando en la interfaz gráfica el problema detectado.

Este test tiene como precondición que el DUT ya se encuentre en modo test, por lo que anteriormente se tiene que haber finalizado un test con resultado de PASS. Además, otra precondición es que la IUT implemente el mecanismo de OTAA y esto se asegura seleccionando la ejecución de este test solo para este tipo de DUT.

A continuación se describe lo que sucede en cada paso:

- Paso 1; el test comienza y espera recibir un mensaje TAOK en el puerto 224 proveniente del DUT para comprobar que el contador de downlink se corresponde con lo esperado. Como respuesta se envía un estímulo al DUT: un mensaje de Session Update.

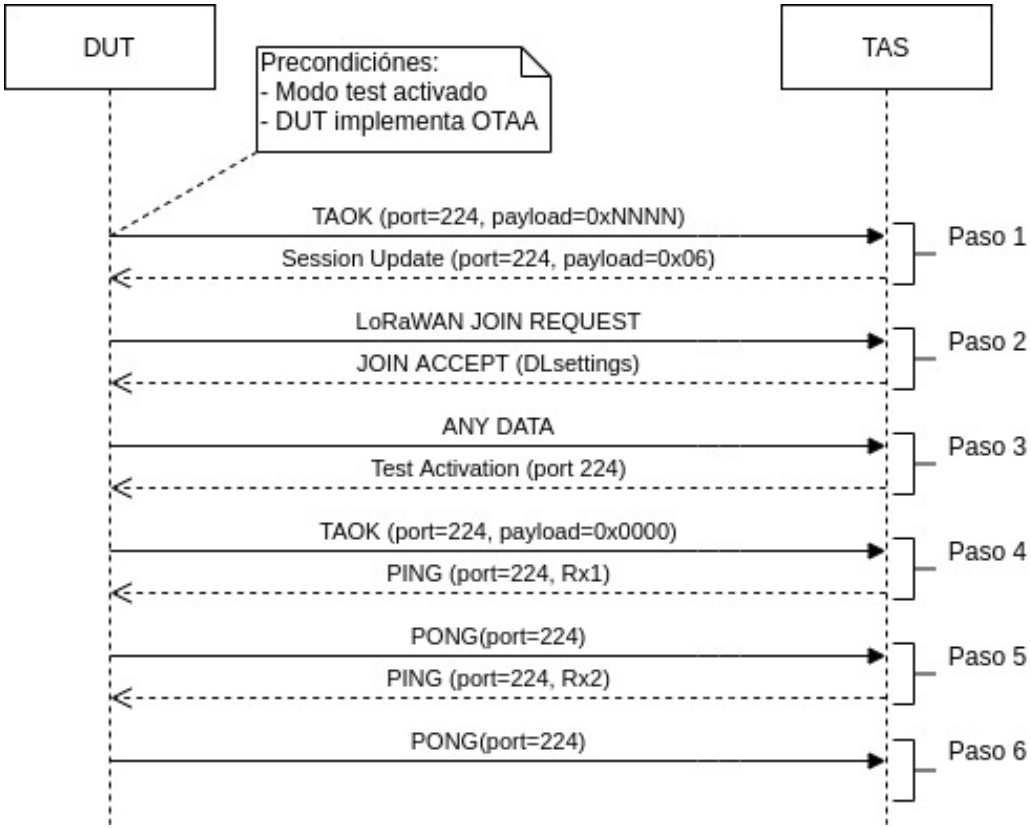

Figura 3.10: Diagrama de comunicación entre el DUT y el TAS para el test ACT 02.

- Paso 2; se espera recibir el mensaje LoRaWAN *JOIN Request* y se responde generando el mensaje *JOIN Accept* correspondiente, configurando con el campo de DLSettings parámetros de DR para las ventanas de downlink (RX1DRoffset=2 y RX2DR=3).
- Paso 3; aquí se espera que el DUT haya recibido el mensaje de *JOIN Accept* y por lo tanto pase a enviar un mensaje de datos (puerto diferente al 224, ya que el DUT no está más en modo test). Se responde a este mensaje con una nueva activación del modo test.
- Paso 4; se espera recibir un nuevo mensaje de TAOK, ahora con el contador de downlink nuevamente en 0. Como respuesta a este mensaje se envía un nuevo estímulo hacia el DUT: un mensaje de PING usando la ventana de downlink RX1 (con el DR de acuerdo a lo configurado en el paso 2).
- Paso 5; este paso verifica que siguiente mensaje que envíe el DUT sea el PONG correspondiente al PING enviado en el paso anterior. Luego se envía un nuevo mensaje de tipo PING pero ahora usando la ventana de downlink RX2.
- Paso 6; finalmente el último paso del test verifica que el mensaje de PING

anterior haya sido procesado correctamente por el DUT y da el test por finalizado. Al no enviarse ningún otro mensaje hacia el DUT, este queda configurado en Modo Test y de acuerdo al protocolo deberá enviar un mensaje de TAOK que será recibido por el primer paso del siguiente Test.

Todos los tests implementados siguen este formato para encadenar las sucesivas verificaciones de distintas funcionalidades de LoRaWAN. El último test de la sesión siempre es el de desactivación, *td_lorawan_deactivate*, que envía el mensaje para notificar al DUT que debe pasar a modo de funcionamiento normal.

3.6. Resumen y conclusión del capítulo

Este capítulo persentó la arquitectura diseñada para la plataforma de pruebas, explicando cómo es la estructura de los tests y el protocolo utilizado para intercambiar mensajes con los dispositivos. Se describió la certificación oficial que ofrece la LoRa Alliance para entender el contexto donde la plataforma de pruebas diseñada ayuda a desarrolladores a verificar que se cumple con las especificaciones del estándar LoRaWAN antes de comenzar este proceso. Adicionalmente se mostró la lógica y secuencia de pasos en el diseño de los tests analizando un ejemplo particular de uno de los tests implementados.

El diseño incluye un servidor de tests, TAS, una interfaz de usuario y un Agente de usuario que oficia de conector entre la plataforma de tests y los componentes proporcionados por el usuario (DUT y LoRa Gateway).

El siguiente capítulo muestra detalladamente cómo se implementó el mencionado diseño en diferentes servicios y módulos de software, fundamentando las tecnologías utilizadas para la comunicación entre los distintos componentes así como para su despliegue.

Capítulo 4

Implementación de la Plataforma

Un factor importante en la implementación de la plataforma es que esta sea modular y extensible, dando facilidad para incorporar nuevos servicios y componentes, pudiéndose integrar con el menor número de modificaciones posibles dentro de otros entornos de pruebas. En las siguientes secciones de este capítulo se describen las tecnologías y criterios de diseño utilizados para lograr este objetivo.

El capítulo comienza con una sección dedicada a mostrar la estructura general de la plataforma de pruebas implementada, detallando cómo los distintos componentes se han dividido en módulos y paquetes para agruparlos según su funcionalidad, facilitando de esta forma el mantenimiento del código y haciendo a la plataforma más fácil de comprender y extender con nuevos tests. Luego la Sección 4.2 explica cómo se implementan los distintos servicios, presentando la arquitectura seleccionada para la comunicación de forma que estos estén desacoplados y se facilite la integración con otros sistemas. El capítulo continúa luego con los detalles de la implementación del componente central de la plataforma, el TAS (Test Application Server), indicando cómo este componente se comunica con el Agente de usuario para lograr un diseño que puede adaptarse a los distintos escenarios que se describieron en la Sección 3.3.1. Para finalizar se incluye una sección presentando la Interfaz de Usuario implementada y otra donde se explica cómo se realiza el despliegue de una instancia de la plataforma de pruebas.

4.1. Estructura general y tecnologías utilizadas

Para la implementación del diseño presentado en el Capítulo 3 se utilizaron, entre otras tecnologías, Python3, Docker y RabbitMQ, y se organizó el desarrollo en los paquetes y módulos mostrados en la Fig. 4.1.

Se implementó un paquete llamado *conformance_testing* que proporciona la estructura básica de los tests explicada en la Sección 3.3. Los componentes implementados en este paquete son independientes del protocolo de comunicación que se esté probando o del tipo de tests que se estén ejecutando.

El paquete llamado *lorawan* incluye la implementación de los tests de conformidad de LoRaWAN reutilizando las funcionalidades del paquete *conforman-*

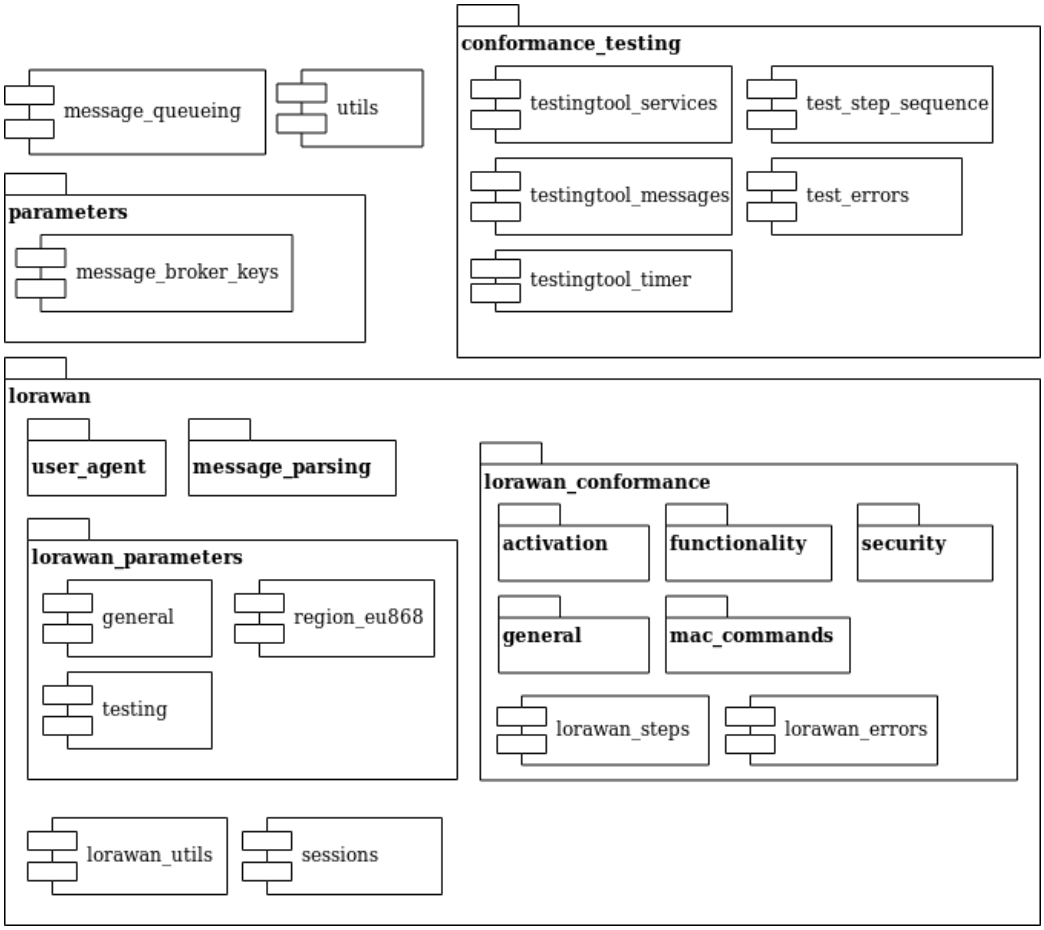

Figura 4.1: Organización del código en paquetes y módulos.

ce_testing y además define los módulos para analizar el formato de los mensajes (*message_parsing*), el agente de usuario (*user_agent*) y los parámetros específicos de LoRaWAN separados por región (*lorawan_parameters*). Si bien se desarrollaron únicamente tests para la región EU 868, se implementa la plataforma pensando en que la misma sea fácilmente extensible para implementar más tests e incluir otras regiones.

4.2. Implementación de servicios

La plataforma de tests se implementa como un conjunto de servicios distribuidos, con componentes independientes que podrían utilizar diferentes tecnologías o estar desarrollados en diferentes lenguajes de programación. Un factor fundamental para adoptar este tipo de diseño es facilitar la integración de las herramientas desarrolladas dentro otros entornos de test, facilitando el despliegue de la plataforma de tests de conformidad para LoRaWAN como un servicio más.

4.2.1. Desarrollo basado en containers

Una tecnología ampliamente utilizada para el desarrollo y despliegue de aplicaciones es Docker [1], herramienta de código abierto que permite encapsular aplicaciones dentro de contenedores de software. Se presenta como una alternativa a la utilización de máquinas virtuales y a diferencia de ellas utiliza menos recursos y evita algunos inconvenientes relacionados a la inicialización y mantenimiento de estas.

Un contenedor es un proceso ejecutándose encapsulado con bibliotecas de software y la cantidad mínima de recursos necesarios para funcionar, aislado adecuadamente del entorno en el que se ejecuta y de otros contenedores. Es además fácilmente replicable, pudiéndose instanciar nuevas copias de este proceso basadas en imágenes que definen el código, dependencias, bibliotecas y todos los recursos que los contenedores requieren para funcionar. A diferencia de cuando se utilizan máquinas virtuales, en donde cada una de estas ejecuta una instancia completa de un sistema operativo, los contenedores solamente encapsulan las dependencias y se aíslan del entorno, utilizando directamente los recursos del sistema operativo en el que se ejecutan.

Para poder crear contenedores y ejecutarlos es necesario instalar el *Docker Engine*, un conjunto de aplicaciones y herramientas que proporcionan el entorno de desarrollo necesario para crear y gestionar imágenes y contenedores. El *Docker Engine* tiene una arquitectura de cliente-servidor, con un servidor (*docker deamon*) encargado de crear imágenes e instanciar y conectar contenedores, y un cliente que le envía peticiones a través de una API (Application Programming Interface). La Fig. 4.2, obtenida de la documentación web de Docker [2], muestra esquemáticamente la arquitectura y la relación entre los componentes definidos por Docker mencionados anteriormente, donde el *docker deamon* responde a peticiones de un cliente Docker para crear contenedores a partir de ficheros *Dockerfile* que definen qué debe incluirse en los contenedores que se instancian a partir de ella. Adicionalmente, es posible descargar imágenes desde repositorios remotos que se utilizan como base para nuevos desarrollos agregando componentes y código a los contenedores que de ella se derivan.

Para cada uno de los servicios definidos en la plataforma de pruebas se crea un fichero *Dockerfile* que especifica una imagen base (i.e. ubuntu) y se le agregan capas con las bibliotecas y herramientas necesarias y se copia el código que debe ejecutarse. Esta imagen luego puede ser compartida, por ejemplo siendo subida a otro repositorio, para que contenedores derivados de ella puedan ejecutarse e integrarse en diferentes entornos. Debido a que todos los requerimientos están incluidos y encapsulados dentro del contenedor se evitan problemas de compatibilidad y se facilita el despliegue de las aplicaciones.

En el Apéndice B se incluyen los ficheros de configuración utilizados para la creación de las imágenes de cada uno de los servicios. También se incluyen los

[1] `https://docs.docker.com/`

[2] `https://docs.docker.com/get-started/overview/#docker\protect\discretionary{\char\hyphenchar\font}{}{}architecture`

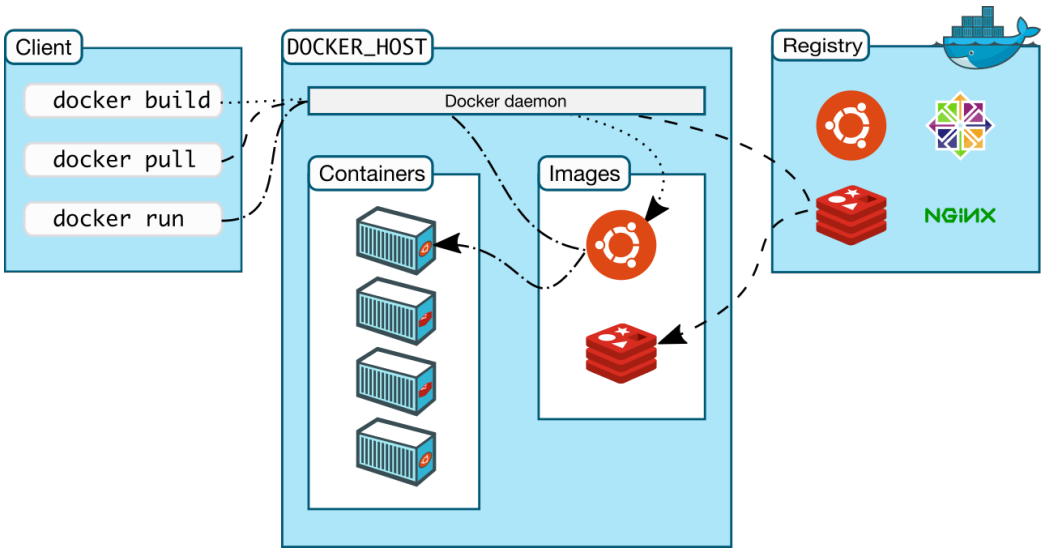

Figura 4.2: Arquitectura y componentes definidos por Docker.

ficheros utilizados para el despliegue de la plataforma, aspecto que se cubre en la Sección 4.6.

4.2.2. Comunicación entre los servicios

En esta sección se describen los mecanismos para comunicar los diferentes servicios definidos en el diseño de la plataforma, explicando el funcionamiento básico de las tecnologías utilizadas.

Con el objetivo de mantener los servicios que interactúan lo más desacoplados posible se implementa la comunicación entre ellos mediante un broker de mensajes. Este tipo de arquitectura proporciona un método asíncrono y escalable para comunicar diferentes aplicaciones y mantenerlas desacopladas.

La idea de implementar la comunicación con esta topología es que un componente central reciba todos los mensajes publicados y los entregue a los subscriptores que corresponda. De esta forma los procesos o servicios que envían mensajes no dependen ni necesitan información sobre los servicios que los consumirán. Por otra parte, los suscriptores se registran para recibir mensajes de determinado tipo sin preocuparse por quien es que los genera.

Se utilizó RabbitMQ [3], un broker de mensajes de código abierto que soporta el protocolo de comunicación Advanced Messaging Queueing Protocol (AMQP). RabbitMQ proporciona mecanismos de seguridad, enrutamiento de los mensajes y persistencia para evitar pérdida de información. La Fig. 4.3 muestra cómo los diferentes servicios del TAS envían los mensajes al *Message Broker* y este se encarga de dirigirlos a los destinatarios correspondientes.

Para entender cómo se realiza el intercambio de mensajes a través del broker es necesario manejar tres conceptos básicos en el diseño de RabbitMQ: *exchanges*,

[3] `https://www.rabbitmq.com/`

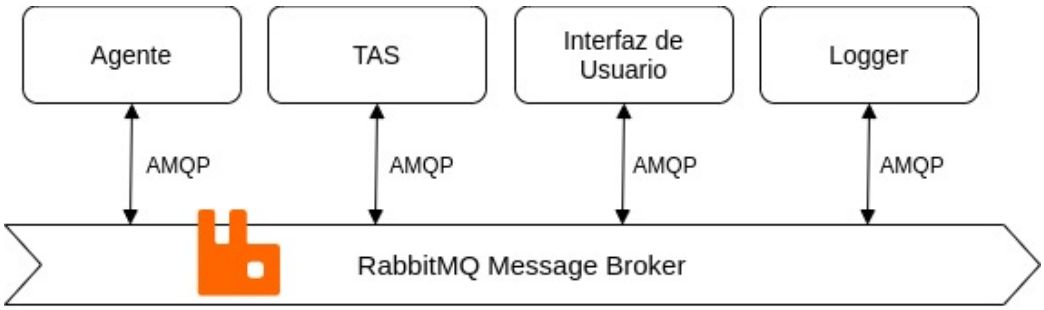

Figura 4.3: Comunicación entre los servicios utilizando AMQP con RabbitMQ.

queues y *bindings*. El exchange es la entidad a donde los productores publican los mensajes, y se encarga de enviarlos a las queues (colas) que tenga asociadas de acuerdo a reglas de ruteo definidas. Los bindings definen cómo son estas reglas de transferencia de mensajes a las distintas queues.

Se tienen los siguientes componentes:

- Productores; que publican mensajes en los exchanges.
- Consumidores; que consumen mensajes desde una queue.
- Queues; que están asociadas a un exchange y son colas que reciben los mensajes de acuerdo a reglas de ruteo definidas.
- Exchanges; entidades que reciben los mensajes y los colocan en las queues que correspondan. De esta forma los productores nunca publican directamente en una queue, y ni siquiera tienen que saber si de hecho existe alguna queue o consumidor. Los exchanges permiten desacoplar la publicación y el consumo de mensajes, siendo estos quienes saben al recibir un mensaje a qué queues debe ser publicado. Entre los distintos tipos de exchanges proporcionados por RabbitMQ, en el presente trabajo se utilizó el de tipo *Topic*. El criterio de este tipo de exchange para definir a quién envía los mensajes recibidos se basa en claves de ruteo (*routing keys*).
- Bindings; son las relaciones entre un exchange y una queue, en las que se especifica una routing key. De esta forma un exchange (de tipo topic) entrega un mensaje a todas las queues para las que hay una correspondencia (match) de la routing key del mensaje con la del binding correspondiente.
- Routing Keys; son palabras separadas por puntos y pueden contener dos caracteres especiales: "*" que indica que se sustituye exactamente una palabra cualquiera en el lugar del asterisco para determinar un match, y "#" que indica que se sustituye cualquier cantidad de palabras en la routing key para comprobar el match.

Por ejemplo, los mensajes con routing key "msg.dev1.up" y "msg.dev2.up" van a ser entregados en una queue que tenga un binding con "msg.*.up", pero no el

mensaje “msg.dev1.down”. Sin embargo, los tres mensajes serán recibidos por una queue con binding “msg.#”.

Un ejemplo de cómo estos elementos son utilizados se muestra en la Fig. 4.4. El *Servicio 1* actúa como productor de mensajes, enviandolos a un *exchange* de tipo topic en el broker de mensajes, sin necesidad de conocer qué otros servicios los reciben. Por otra parte, *Servicio 2* y *Servicio 3* han declarado *queues* en el broker de mensajes y consumen los mensajes que llegan a estas sin necesidad de conocer exactamente qué otro servicio los envía.

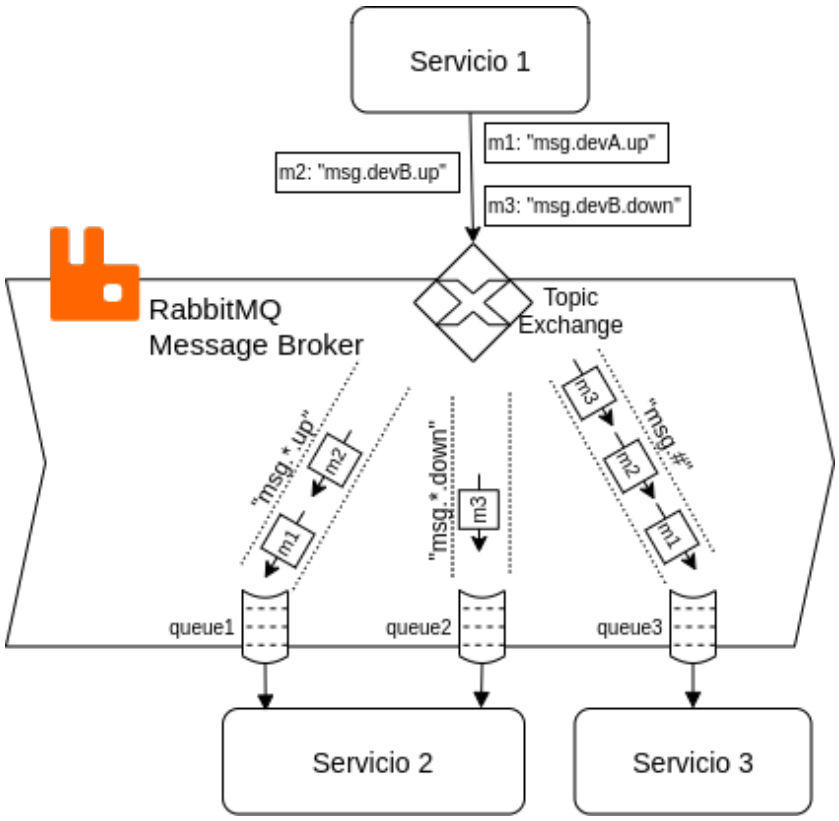

Figura 4.4: Ejemplo de comunicación entre servicios utilizando RabbitMQ.

En la situación del ejemplo estos son las acciones ejecutadas y el rol de cada uno de los componentes involucrados:

- Servicio 1: actúa como productor de mensajes y envía los mensajes *m1*, *m2* y *m3* al broker con las routing keys “msg.devA.up”, “msg.devB.up” y “msg.devB.down”
- Servicio 2: actúa como consumidor de mensajes, declarando las queues queue1 y queue2 en el broker y haciendo un binding de las mismas con las routing keys “msg.*.up” y “msg.*.down” respectivamente.
- Servicio 3: es un consumidor de mensajes en la queue3, que fue declarada y asociada al exchange haciendo un binding con la routing key “msg.#”.

- Topic exchange: Se encarga de hacer el ruteo de los mensajes en base de la routing key que el mensaje tenga asociada. De esta forma decide a que queues debe enviar una copia de cada mensaje recibido.

Módulos de interacción con el message broker

Entre las interacciones con el message broker están la creación de queues, creación de exchanges, publicación de mensajes o el registro de una aplicación para que consuma mensajes desde una queue. Para estas interacciones se utiliza Pika [4], librería que implementa un cliente Python del protocolo AMQP para comunicación con RabbitMQ.

Para mantener el desarrollo lo más independiente posible de la librería elegida para la comunicación con el broker RabbitMQ, se encapsulan las funcionalidades en la clase *MqInterface* representada en la Fig. 4.5 del módulo *message_queueing*.

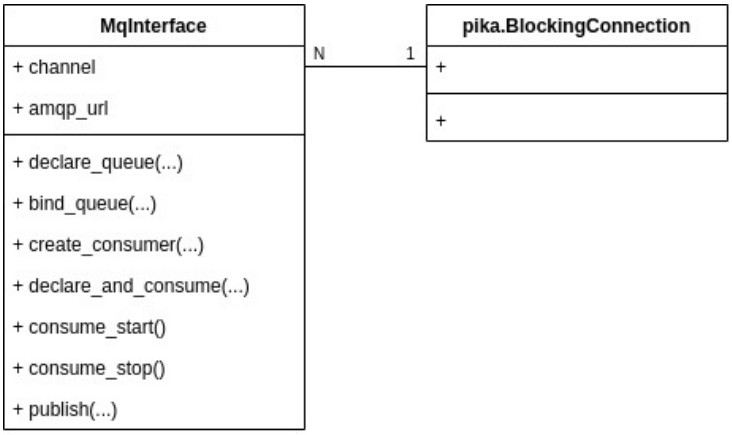

Figura 4.5: Clase que encapsula las funcionalidades de la librería Pika.

Para poder publicar, consumir mensajes, o realizar cualquier interacción con el broker, las aplicaciones deben establecer una conexión TCP con él. Para minimizar el costo computacional de la creación de conexiones, AMQP define canales de comunicación sobre cada conexión. Estos canales pueden visualizarse como conexiones virtuales sobre una conexión AMQP, tal como muestra la Fig. 4.6.

El módulo *message_queueing* mantiene un única conexión con el broker y utiliza un canal para cada instancia de la clase *MqInterface*, restableciendo la conexión en caso de que la misma no esté activa al momento de ser solicitada.

Con la estrategia utilizada para el ruteo de mensajes, implementada con la arquitectura de comunicaciones que ofrece RabbitMQ con AMQP, cualquier servicio puede añadir su propia queue haciendo un binding según los mensajes que quiere consumir. Para organizar los criterios con los que se envían los mensajes entre servicios y facilitar cambios se cuenta con el módulo *message_broker_keys*, definido en el paquete *parameters* introducido en la Fig. 4.1 de la Sección 4.1, que contiene las routing key definidas.

[4] `https://github.com/pika/pika`

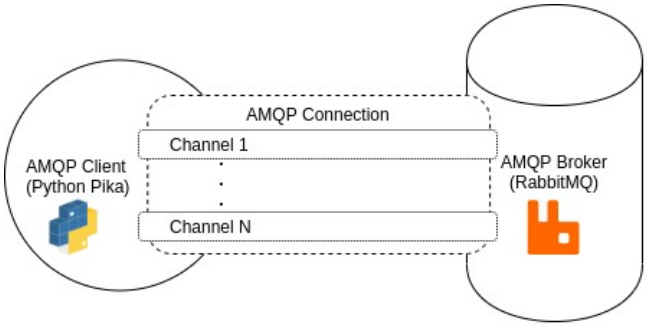

Figura 4.6: Canales sobre una conexión AMQP.

Una vez fundamentada la elección de Docker para implementar los distintos servicios y explicada la forma en la que estos se comunican desacoplando el envío y la recepción de mensajes, la siguiente sección explica la implementación del componente central de la plataforma de pruebas: el Test Application Server.

4.3. Test Application Server

La estructura básica del Test Application Server, componente central de la plataforma y diseñado de acuerdo a lo explicado en la Sección 3.3.1, está implementada en el paquete *conformance_testing*. La Fig. 4.7 muestra los módulos implementados y las dependencias con otros paquetes.

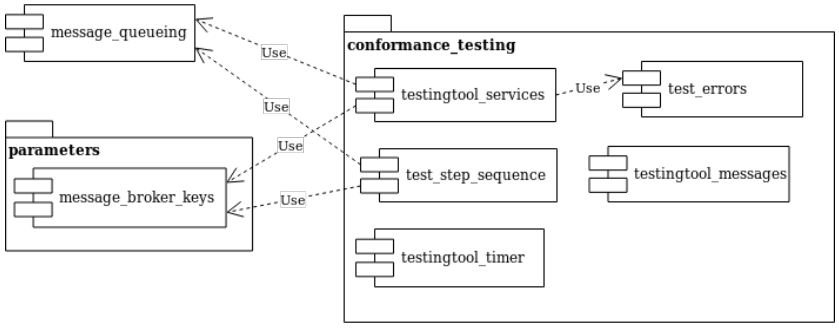

Figura 4.7: Módulos y paquetes que implementan el TAS.

En el módulo *testingtool_services* se encuentra la clase *TestSessionCoordinator*, Test Session Coordinator (TSC), que hereda de *MqInterface* ya que centralizará la comunicación con los demás servicios a través del broker. El *TestSessionCoordinator* es el componente que coordina la ejecución de los tests, mantiene el downlink counter (ver Sección 3.4), y tiene el registro del DUT con los parámetros necesarios para iniciar una sesión LoRaWAN y comunicarse con él. La Fig. 4.8 muestra un diagrama de clases con el *TestSessionCoordinator* implementado en el módulo *testingtool_services*, de acuerdo al diseño de la Fig. 3.8, especificando las relaciones con clases implementadas en otros módulos.

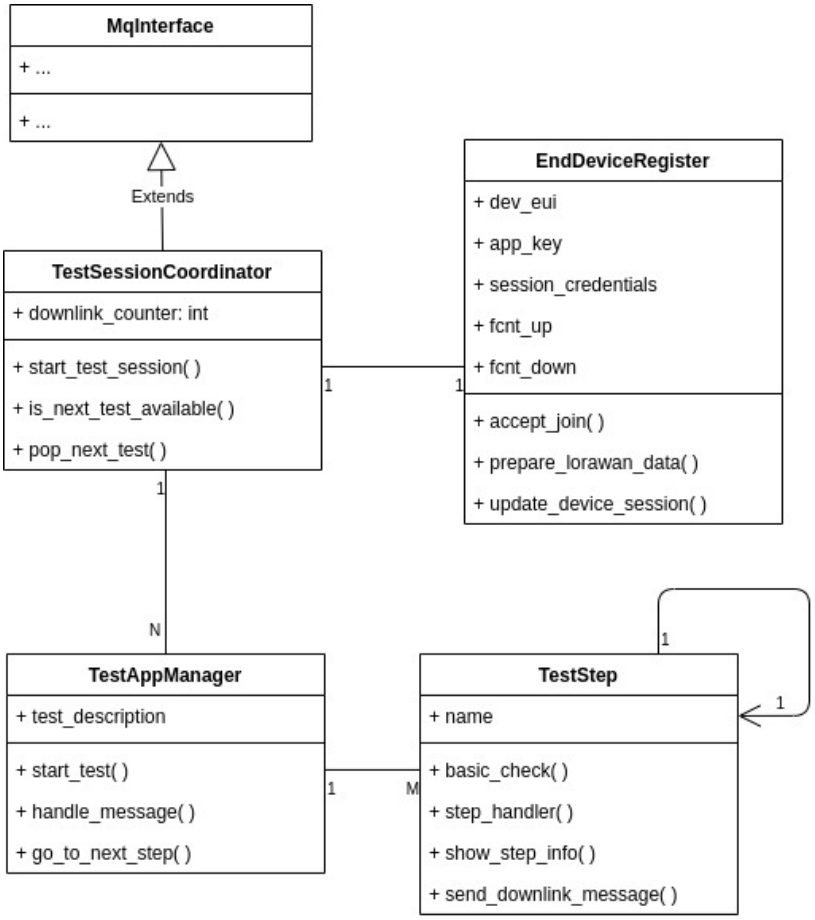

Figura 4.8: Diagrama de clase del TestSessionCoordinator.

La Fig. 4.9 muestra la secuencia de eventos y mensajes intercambiados entre el TSC (*TestSessionCoordinator*), la interfaz de usuario (GUI) y el DUT (a través del LoRa Gateway y el Agente, representados cómo una única entidad).

El módulo principal del TAS instancia un *TestSessionCoordinator* (paso previo no mostrado en el diagrama) que muestra al usuario la configuración necesaria para el Agent (punto A en el diagrama), esta configuración contiene los parámetros del broker de mensajes AMQP utilizado y contempla el escenario donde el RabbitMQ Broker y el TAS se estén ejecutando de forma remota y sea necesario conocer la URL a la que hay que dirigir los mensajes.

Luego de recibir los parámetros de configuración del DUT (credenciales LoRaWAN) y los tests que se desea ejecutar (punto B), el TSC indica a través de la GUI que está todo listo para comenzar (punto C) y queda a la espera de que el usuario prepare el Agent y el DUT.

A partir de ese momento se carga el TestManager correspondiente a cada uno de los tests seleccionados (punto D) y se ejecutan todos los pasos (punto E). Cada test envía información de cada paso ejecutado a la GUI y una vez ejecutados todos los tests se envía el reporte final (punto F).

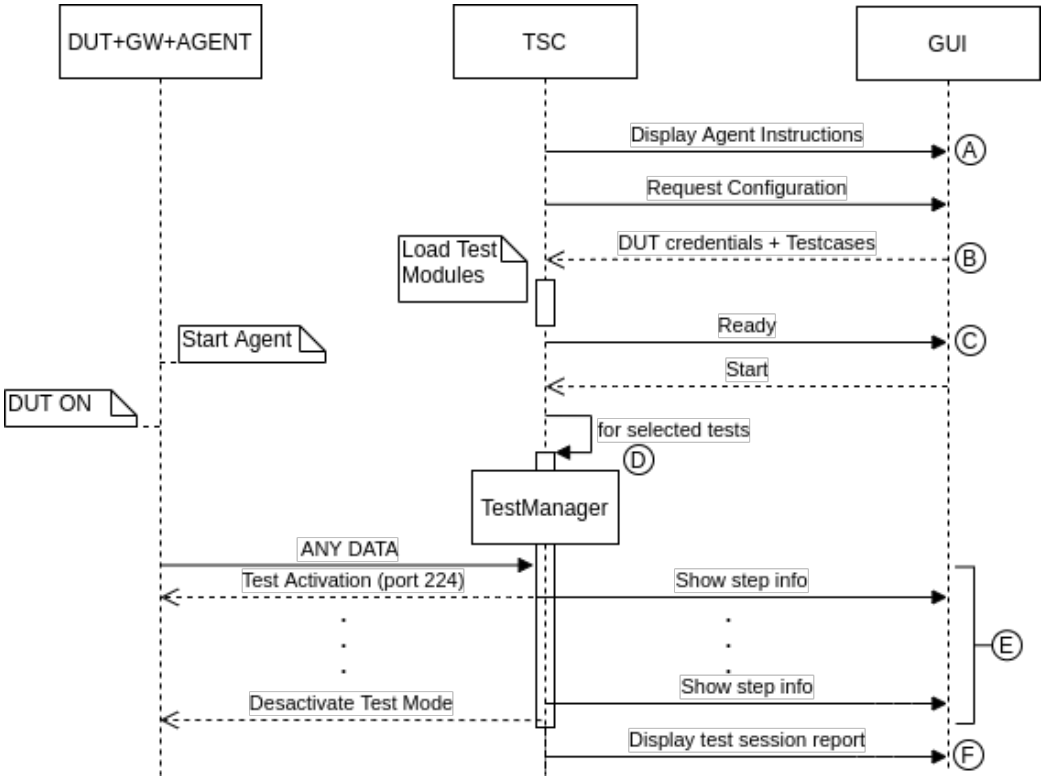

Figura 4.9: Secuencia de eventos y mensajes en una sesión de tests.

El reporte final se inicializa al momento de comenzar la sesión de test y se actualiza cada vez que un nuevo test finaliza. Esta actualización es el mensaje de que ningún error fue detectado (PASS) o con información de error en caso de FAIL. El módulo principal del TAS captura y procesa los errores detectados y los inserta en el reporte en caso de que el resultado de un test sea FAIL.

La siguiente sub-sección explica cómo se implementó el diseño de los tests, mostrando la arquitectura básica de estos que es genérica y no asume nada respecto al protocolo de comunicación que se estará probando. Luego se continúa con una explicación detallada de cómo se implementan los tests para LoRaWAN basándose en un test particular a modo de ejemplo.

4.3.1. Arquitectura básica de los tests

En el módulo *test_step_sequence* se define la estructura central de los tests del TAS. Aquí encontramos dos clases abstractas que definen la interfaz para los *TestManager* y *TestStep*, clases que serán luego implementadas por los diferentes tests. Cada TestManager conoce los pasos que se ejecutarán en el test y en cada paso del test (*TestStep*) se verifica que un aspecto concreto del protocolo se cumple con la especificación de referencia (i.e. LoRaWAN v1.0.3). Para ello la clase *TestStep* define los métodos *basic_check* y *step_handler*, facilitando la reutilización de código mediante herencia, ya que varios test que deben realizar el mismo chequeo básico

pueden heredar de la misma clase y sobreescribir el *step_handler*.

En caso de detectarse un error el *TestStep* correspondiente lanza una excepción. De esta forma se utiliza el sistema de manejo de excepciones de Python para capturar y reportar los problemas detectados en los tests. La Fig. 4.10 muestra los errores definidos en el módulo *conformance_testing.test_errors*, más adelante serán introducidos más errores específicos de LoRaWAN.

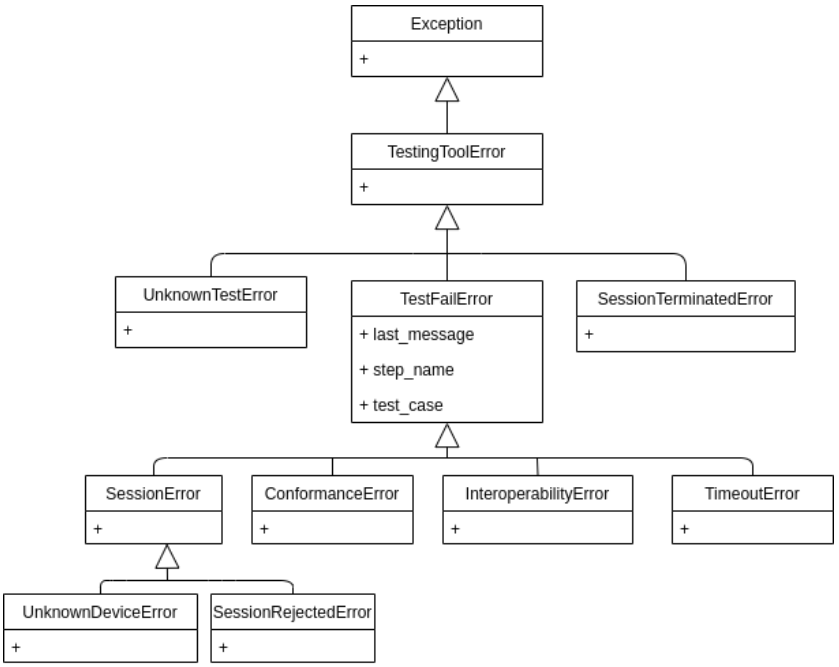

Figura 4.10: Errores definidos para los tests de conformidad.

- TestingToolError: todos los errores definidos en la plataforma de tests heredarán de esta clase base.
- UnknownTestError: este tipo de error indica que se ha solicitado un test no reconocido dentro de los módulos definidos en la plataforma.
- SessionTerminatedError: indica que la sesión de tests se ha interrumpido por solicitud del usuario al enviar una señal de finalización.
- TestFailError: aquí se agrupan todos los errores que indican el fallo de un test, y contiene información del test, el paso que se estaba ejecutando al momento de producirse la excepción y el último mensaje procesado.
- TimeoutError: indica que se ha producido un error por falta de respuesta del DUT dentro de los límites determinados para el paso que se estaba ejecutando.
- SessionError: agrupa los errores producidos en la información de sesión del DUT, se contemplan estas situaciones:

    - UnknownDeviceError: si el DUT que intenta comunicarse con la plataforma de tests no ha sido registrado, y por lo tanto no se cuenta con las claves del mismo.
    - SessionRejectedError: cuando se ha producido un error en el procesamiento de la solicitud de sesión por parte del dispositivo.

- InteroperabilityError: agrupa los errores debido a una respuesta que no era la esperada para ese paso del test.
- ConformanceError: este tipo de error comprende las situaciones donde el formato del mensaje recibido no era el correcto al no cumplir con la especificación de LoRaWAN.

4.3.2. Tests de conformidad para LoRaWAN

El módulo *lorawan_conformance.lorawan_steps* proporciona las clases que componen los tests de LoRaWAN. Cada test que se implementa tendrá definido un *TestAppManager* (que hereda de la clase abstracta *TestManager*) y contiene los pasos correspondientes del test. Cada uno de estos pasos estará implementado como una clase que hereda de *TestStep* y estas se crean de forma de reutilizar las verificaciones definidas por otros tests.

La Fig. 4.11 muestra las clases definidas para los pasos de los tests. Todos los tests heredan de la clase *LorawanStep* que hereda de *TestStep* e implementa el método *basic_check*, en donde se comprueba el MIC del mensaje y se verifica si el mensaje recibido es de tipo CONFIRMED o UNCONFIRMED. Este chequeo básico es utilizado entonces por todos los tests, que luego implementan el método *step_handler* para realizar las verificaciones adicionales que correspondan. La clase *TestStep* también define métodos para enviar hacia la interfaz de usuario información sobre el resultado del paso ejecutado (*show_step_info(...)* y para enviar mensajes de downlink hacia el DUT (*send_downlink_message(...)*).

Las principales clases definidas son,

- LorawanStep: clase base para todos los tests de LoRaWAN en la que se verifica que el MIC sea correcto.
- JoinRequestHandlerStep; en esta clase se implementa el manejo del mensaje de LoRaWAN JOIN REQUEST por parte del DUT, verificando el formato del mensaje recibido y enviando el JOIN ACCEPT para confirmar el inicio de sesión.
- WaitDataToActivate: este paso espera recibir un mensaje de datos por parte del DUT para activar el Modo Test.
- WaitActokStep: espera recibir un mensaje de TAOK y verifica el contador de downlink.
- WaitPong; luego de haber sido enviado un mensaje de tipo PING al DUT, este paso del test está en espera de la respuesta para verificar que la misma es correcta.

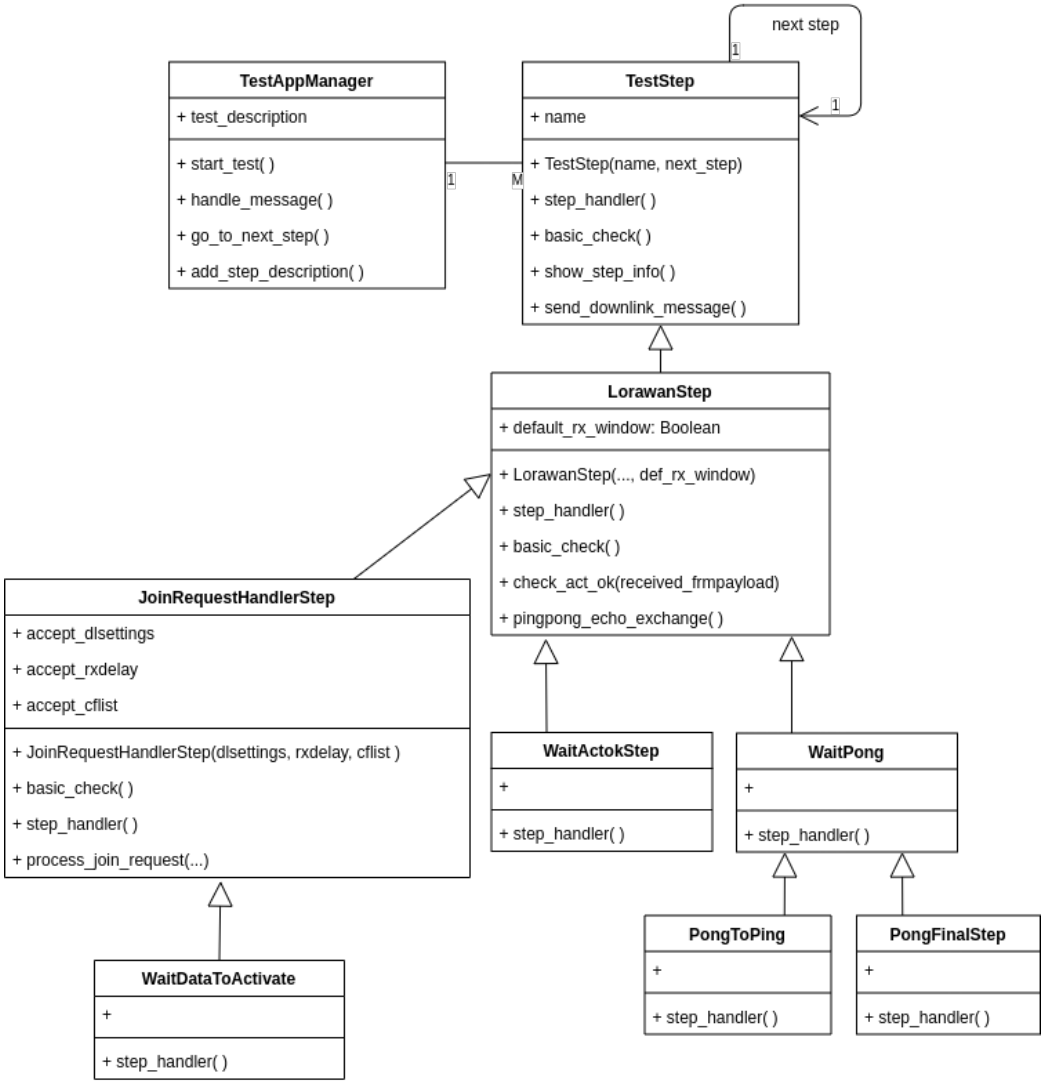

Figura 4.11: Pasos definidos para los tests de conformidad de LoRaWAN.

- PongToPing: esta clase hereda de WaitPong, reutilizando el código que verifica el mensaje de PONG, para luego de comprobar que es correcto enviar un nuevo mensaje PING.

- PongFinalStep: este paso se utiliza para finalizar el test, una vez comprobado que la verificación del mensaje PONG es correcta. Al finalizar se deja de consumir mensajes y se devuelve el control al TSC para que continúe con el siguiente test.

4.3.3. Implementación de los Tests de Conformidad

En la Sección 3.5 del Capítulo 3 se describió el diseño de un test de conformidad que comprueba el mecanismo de activación OTAA del DUT; ahora se mostrará la implementación del mismo indicando cómo se reutiliza el código en los diferentes tests.

El test *td_lorawan_act_02* tiene como objetivo probar que la activación de tipo OTAA está correctamente implementada. Cambiando los parámetros de DR de las ventanas de recepción de downlink (RX windows) se verifica que el DUT puede unirse a una red LoRaWAN usando OTAA en Rx1 y en Rx2. Este test asume que el DUT está configurado en Modo Test y soporta el mecanismo de OTAA.

Se crea un módulo Python para cada test, en este caso nos estamos enfocando en el módulo *lorawan_conformance.activation.td_lorawan_act_02*. Este módulo define el *TestAppManager* del test donde se especifican los pasos que deben ejecutarse como instancias de las clases creadas previamente. En la Fig. 4.12 se puede apreciar la secuencia de pasos y la relación de herencia entre algunas de ellas así como de pasos definidos previamente.

El diagrama de la Fig. 4.13 muestra de forma gráfica el flujo de mensajes esperado para el test ACT 02.

- Paso 1: el test comienza con un paso de la clase *ActokToTriggerJoin* que luego de verificar el mensaje de TAOK, reutilizando el código de *WaitActokStep*, envía un mensaje de Session Update para indicar al DUT que debe enviar un LoRaWAN JOIN Request.
- Paso 2: el segundo paso es un *JoinRequestHandlerStep* que espera recibir el JOIN REQUEST solicitado en el paso anterior para enviar un JOIN Accept configurando parámetros para las ventanas de downlink Rx1DRoffset=1 y Rx2DR=3.
- Paso 3: el tercer paso, de la clase *WaitDataToActivate*, espera recibir un mensaje de datos para activar nuevamente el modo test en el DUT.
- Paso 4: en este paso, instancia de la clase *ActokToPing*, se espera recibir un TAOK para verificarlo y enviar un mensaje de PING al DUT utilizando la ventana de downlink Rx1. Al generar el mensaje de PING este paso configura un atributo de la instancia del paso siguiente (paso 5) con los bytes que se espera recibir, proceso que utiliza el método *pingpong_echo_exchange* de la clase *LorawanStep* mostrado en la Fig. 4.12.
- Paso 5: el quinto paso del test, instancia de la clase *PongToPing*, espera el mensaje PONG y verifica que el mismo se corresponda al enviado anteriormente. Luego se genera un nuevo mensaje de PING pero ahora se envía utilizando la ventana de downlink Rx2.
- Paso 6: el último paso del test es una instancia de la clase *PongFinalStep*, que luego de verificar el mensaje de PONG recibido (de acuerdo a lo configurado por el paso anterior, de forma análoga a lo sucedido en el paso 4) da el test por finalizado con un resultado de PASS.

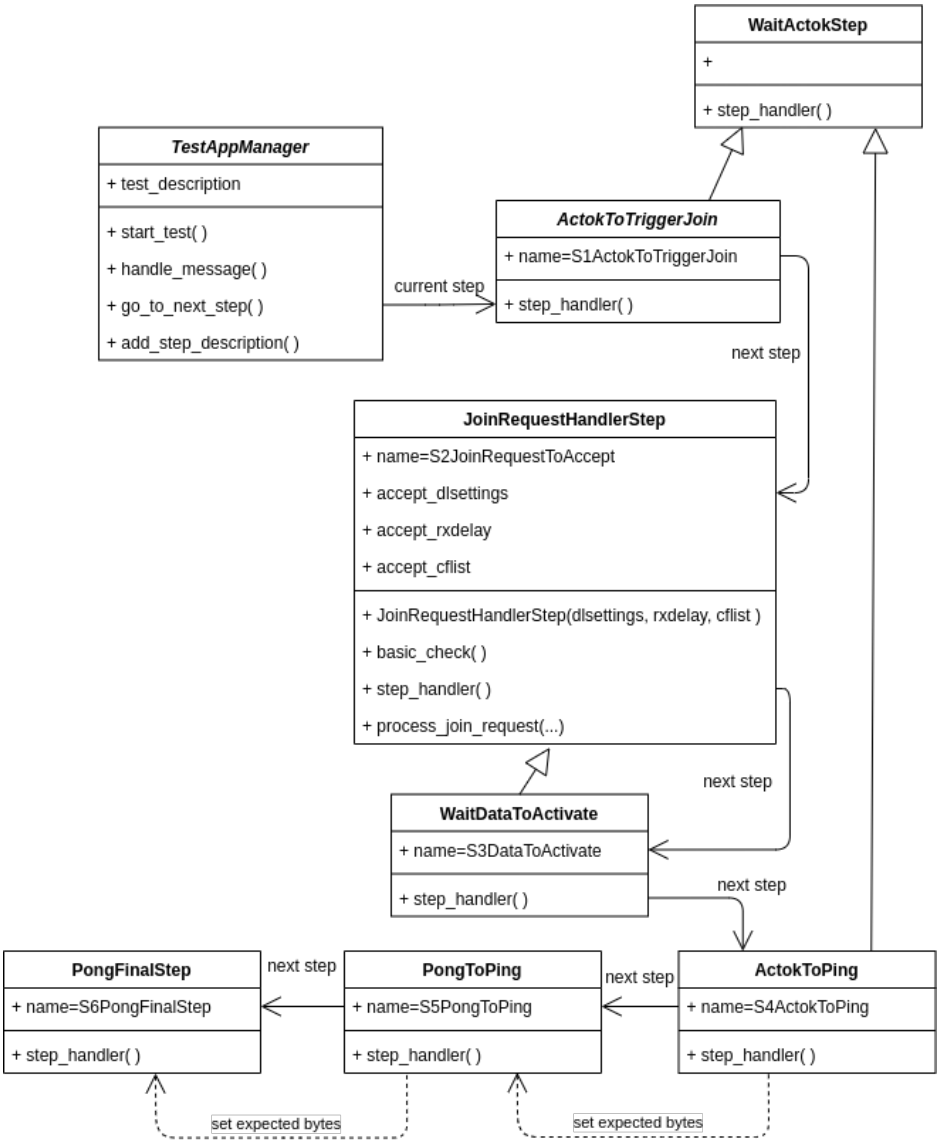

Figura 4.12: Diagrama de clases de la implementación del tests ACT 02.

Si durante la ejecución de alguno de los pasos definidos en el test se detecta algún error se lanza un excepción que deberá ser atrapada por el módulo principal del TAS para incluir la información del error en el reporte de los tests que se muestra al usuario. En la Fig. 4.14 se pueden ver los errores definidos para los tests the LoRaWAN.

A partir de estos errores se derivan otros para poder especificar con mayor granularidad el tipo de error detectado. Si un nuevo test desea especificar una condición de error particular simplemente debe extender alguna de estas clases. El siguiente listado muestra los principales tipos de errores implementados:

- MessageFormatError, agrupa todos los errores de format en los mensajes

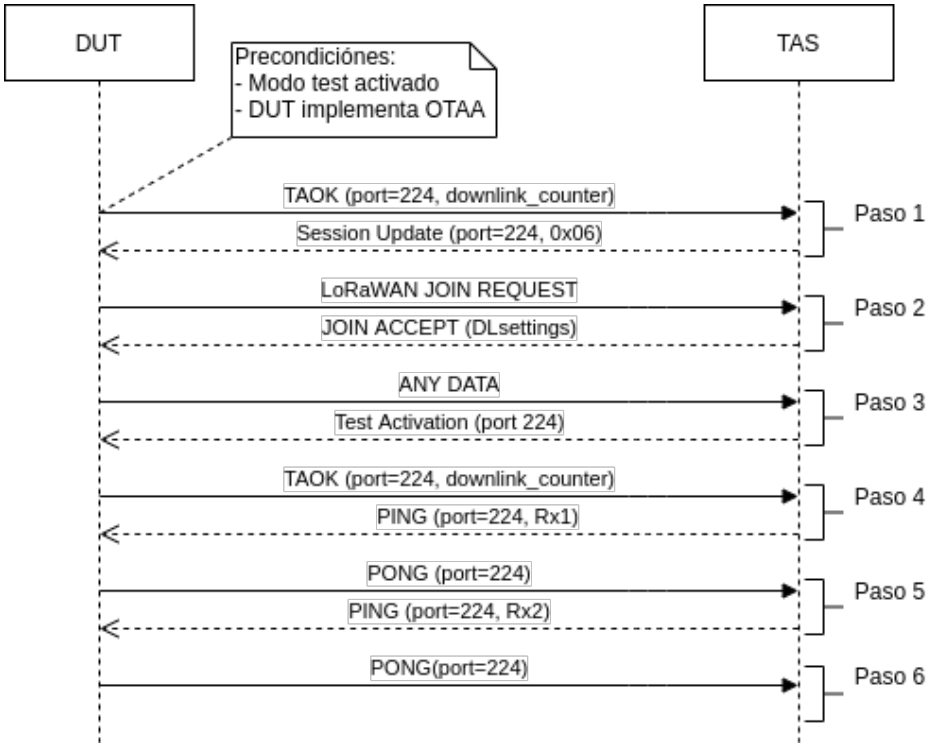

Figura 4.13: Diagrama de comunicación entre el DUT y el TAS para el test ACT 02.

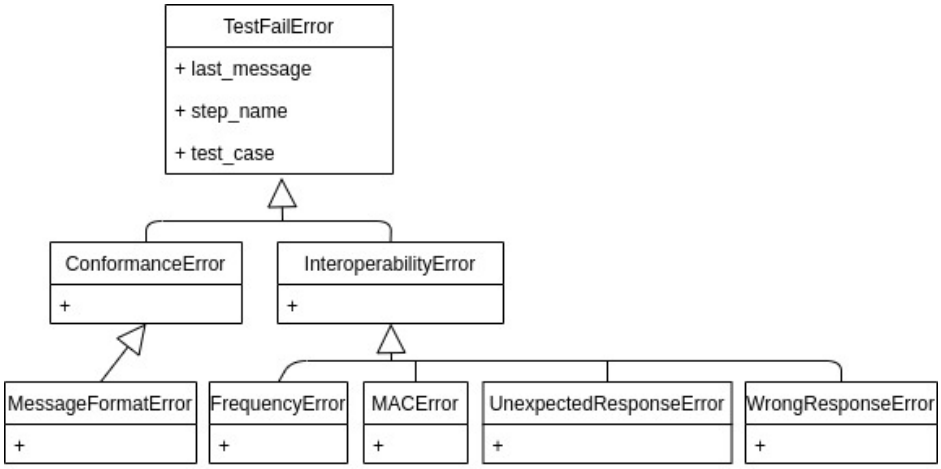

Figura 4.14: Principales excepciones definidas para identificar errores de los tests de LoRaWAN.

LoRaWAN:

- MACPayloadError
  - FHDRError
  - FPortError
  - FRMPayloadError

        - JoinRequestError
        - MACPiggibackedAndPort0, usado cuando se detecta un mensaje con comandos MAC en el FRMPayload (usando puerto 0) y en el frame header (FOpts del FHDR).
      - MHDRError
      - MICError
  - WrongResponseError, indica que el contenido del mensaje no era el esperado:
      - EchoError, usado cuando el mensaje de PONG recibido no se corresponde al último PING enviado.
      - ActokCounterError, para notificar que hay un error en el contador de downlink respecto al registro del TAS.
  - MACError, agrupa todos los errores mientras se envían comandos MAC al DUT:
      - NoMACResponseError, usado cuando no se recibe la respuesta a un comando MAC enviado previamente.
      - WrongMACFormatError.

Habiendo explicado la arquitectura del TAS y cómo se implementan los tests, la siguiente sección describe el componente que se encarga de enviar hacia el TAS todos los mensajes provenientes del DUT: el Agente de Usuario.

4.4. Agente de Usuario

El Agente de usuario es el servicio que hace de puente entre el LoRa Gateway y el TAS. Para ello debe comunicarse con el Gateway utilizando una conexión UDP con el protocolo definido por Semtech para el Packet Forwarder (PF) [5] y establecer una conexión segura con el TAS mediante AMQP.

4.4.1. Semtech Packet Forwarder Protocol

El PF es un proceso que proporciona una forma simple de conectar el concentrador LoRa de un Gateway con una red LoRaWAN. Para ello define un protocolo que permite enviar los mensajes de uplink provenientes del concentrador, agregando metadata e información de estado del Gateway, y recibe los mensajes de downlink provenientes de la red LoRaWAN para agendar su transmisión por LoRa.

Si bien este protocolo no contempla aspectos básicos de seguridad, no representa un problema para este caso de uso ya que la comunicación será solamente en una red interna entre el Gateway y el PC donde se ejecuta el Agente. Todos los

[5] `https://github.com/Lora-net/packet_forwarder`

mensajes provenientes del LoRa Gateway son encapsulados y enviados de forma segura por AMQP hacia el Broker de mensajes que conecta con el TAS. El Semtech Packet Forwarder es un protocolo ampliamente utilizado por distintos fabricantes de Gateways y soportado en redes LoRaWAN de uso masivo como por ejemplo The Things Network [6].

Se definen los siguientes tipos de mensajes:

- PUSH_DATA: utilizado para enviar mensajes de uplink desde un Gateway LoRa hacia la red LoRaWAN. Está compuesto por un token, la dirección MAC del Gateway y un objeto JSON con un conjunto de N paquetes recibidos por el Gateway. N es variable y depende del número de paquetes recibidos por el Gateway.

- PUSH_ACK: se utilizan para reconocer la recepción de un mensaje de PUSH_DATA, y contiene el mismo token para identificar a qué mensaje se refiere.

- PULL_DATA: este tipo de mensaje es utilizado por un Gateway para solicitar datos de downlink. Se envía de forma periódica para asegurar que hay una ruta disponible para los mensajes de downlink.

- PULL_ACK: el servidor LoRaWAN debe enviar este mensaje de reconocimiento para confirmar que el canal para mensajes de downlink está abierto y puede comenzar a enviar mensajes de tipo PULL_RESP.

- PULL_RESP: son mensajes de downlink enviados por el servidor LoRaWAN, los cuales se pueden mandar una vez establecida la conexión mediante el envío del primer PULL_DATA. Estos mensajes contienen un token que los identifica y un objeto JSON con los mensajes que se desea agendar para ser enviados por el Gateway.

- TX_ACK: el Gateway envía este mensaje de reconocimiento de los mensajes de downlink recibidos y puede incluir códigos de error, por ejemplo:

    - TOO_LATE, si ya es tarde para agendar un mensaje en el timestamp solicitado.
    - TOO_EARLY, cuando el timestamp del paquete a agendar para downlink es mayor al admitido: es demasiado pronto para agendar un mensaje en ese momento.
    - COLLISION_PACKET, si hay ya un paquete agendado para enviarse en el mismo instante.
    - TX_FREQ, si se ha solicitado enviar el mensaje en una frecuencia no soportada.

[6] `https://www.thethingsnetwork.org/docs/gateways/packet-forwarder/semtech-udp.html`

4.4.2. Estructura del Agente

El servicio del Agente y las funcionalidades asociadas se definen en el paquete *lorawan.user_agent.bridge* que define los módulos:

- *bridge_main*: módulo principal que instancia e inicia al Agente.
- *agent_bridge*: define la clase *SPFBridge* que implementa la comunicación con el broker AMQP. Además, también maneja la comunicación con el Gateway a través de una conexión UDP usando una instancia de la clase *UDPListener*, que se describe a continuación.
- *udp_listener*: este módulo implementa la concurrencia entre el envío de mensajes de uplink y downlink. Define la clase *UDPListener*, un *Thread* que maneja la conexión UDP con el Gateway de forma concurrente a la comunicación con el TAS que se realiza mediante AMQP utilizando el broker de mensajes.

En la Fig. 4.15 se muestra en un diagrama de clases los métodos y atributos definidos por estas clases.

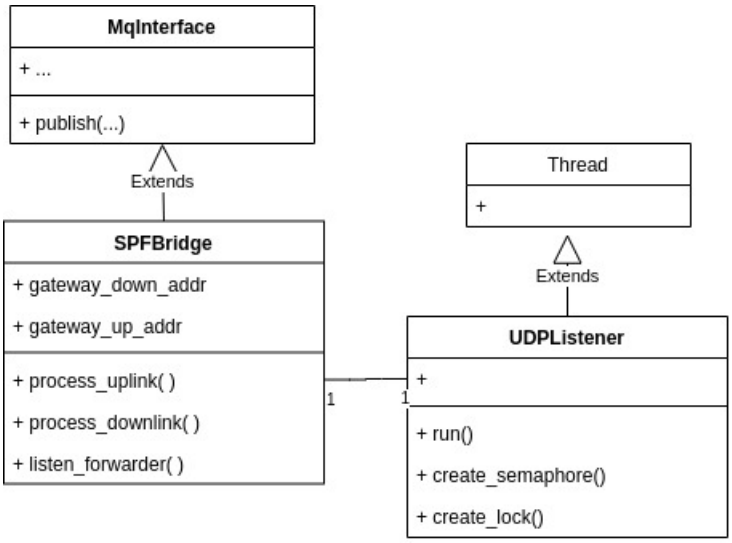

Figura 4.15: Diagrama de las clases que implementan el Agente.

El *UDPListener* encapsula la ejecución concurrente del consumo de mensajes de downlink proveniente del TAS y el procesamiento de los mensajes UDP de uplink provenientes del Gateway. Los módulos y clases que lo utilizan no se ven afectados si por ejemplo se decidiera cambiar la forma de concurrencia de Multithreading a Multiprocessing.

Se utilizan semáforos y locks para asegurar que el acceso a los recursos compartidos por el código que maneja el tráfico de downlink y uplink se hace de forma coordinada. Además, no se comienza a consumir mensajes de downlink hasta que

no se ha recibido el primer mensaje de PULL_DATA desde el Gateway y el canal de downlink está por lo tanto activo. En la Fig. 4.16 se muestra de forma esquemática la secuencia de eventos y mensajes que se intercambian entre el PF que se ejecuta en el LoRa Gateway (GW), el Agente y el TAS (a través del broker AMQP). En esta figura se incluye al DUT dentro del bloque DUT+GW+PF, y son sus mensajes los que serán enviados dentro de los PUSH_DATA que envía el PF hacia el Agent Bridge.

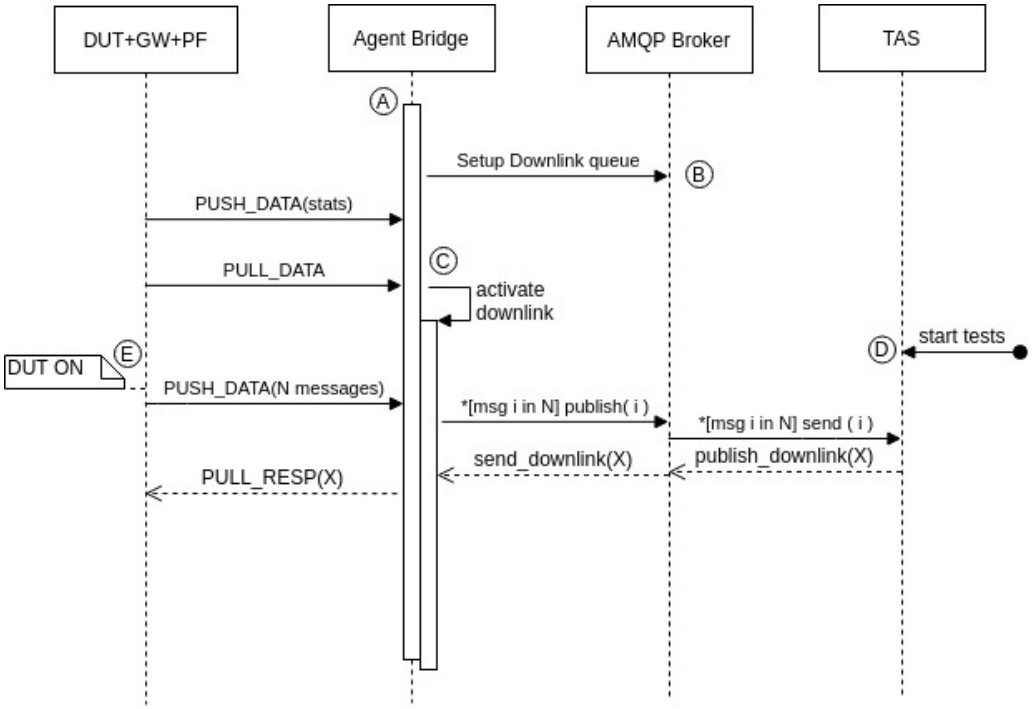

Figura 4.16: Secuencia de mensajes y eventos en la ejecución del Agente.

Estos son los eventos que ocurren a partir de que se inicia el Agente, justo después de recibir la configuración a través de la GUI, tal como se mostró en la Fig. 4.9 de la Sección 4.3.

- Punto A: Se inicia el Agente una vez conectado y configurado el Gateway. El usuario conoce la configuración del broker AMQP, información disponible a través de la GUI luego de iniciado el TAS.
- Punto B: El Agente crea una queue para los mensajes de downlink en el Broker pero todavía sin subscribirse a los mensajes que llegan a esta.
- Punto C: El *UDPListener* comienza a analizar los mensajes que llegan desde el PF y luego de recibir un PULL_DATA configura los parámetros de downlink e indica mediante la liberación de un semáforo que se puede comenzar la recepción de mensajes de downlink provenientes del TAS.
- Punto D: Luego de que el Agente está configurado y listo el usuario puede dar la señal de comienzo de los tests en la GUI.

- Punto E: Una vez iniciada la sesión de tests se enciende el DUT para que comience a interactuar con el TAS siguiendo el protocolo de test.

4.5. Interfaz de Usuario

En la presente Sección se explica cómo la Interfaz de Usuario fue implementada, mostrando la estructura de la misma con los componentes que se encargan de presentar información al usuario así como aquellos encargados de procesar los comandos y respuestas a solicitudes que el usuario envía al TAS.

La interfaz de usuario tiene las siguientes funciones:

- mostrar los mensajes que el TAS dirige el usuario,
- permitir configurar la sesión de test eligiendo las pruebas que se desea ejecutar,
- configurar las credenciales LoRaWAN con las que se desea configurar el DUT,
- enviar la señal del comienzo de tests cuando el usuario tiene el Agent configurado,
- mostrar el reporte final con el resultado de los tests.

Estas funcionalidades son proporcionadas por un conjunto de módulos y servicios implementados en el paquete *user_interface* representado en la Fig. 4.17.

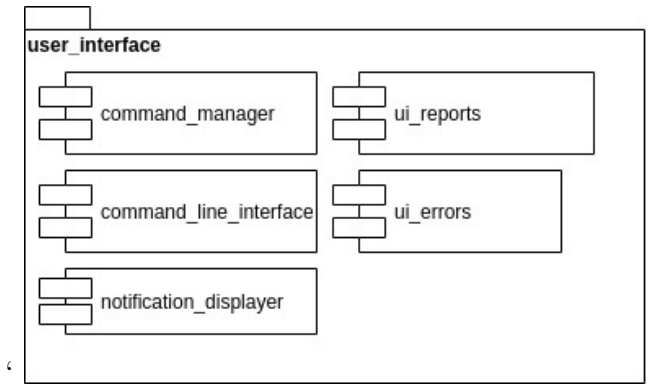

Figura 4.17: Módulos que implementan las funcionalidades de la interfaz de usuario.

Los módulos *command_manager* y *notification_displayer* forman la base de la interfaz de usuario, sobre la que luego pueden agregarse más servicios, cómo por ejemplo una interfaz web. La Fig. 4.18 muestra cómo estos servicios interactúan con el TAS y son la base para luego construir otras formas de visualizar la información y enviar comandos. El servicio del CLI (Command Line Interface) del módulo *user_interface.command_line_interface* permite enviar comandos al TAS sin necesidad de contar con un Web Front End u otro tipo de interfaz gráfica (fuera del alcance de este trabajo de tesis).

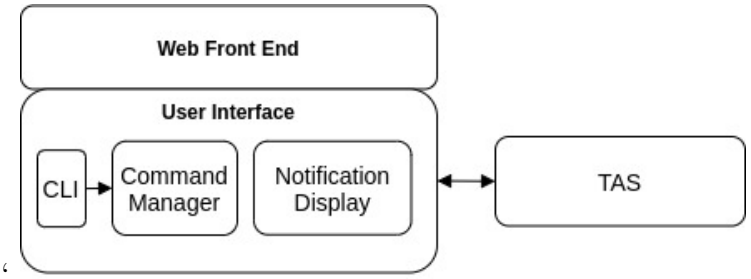

Figura 4.18: Servicios de la interfaz de usuario interactuando con el TAS.

El servicio de *Notification Display* se encarga de recibir todos los mensajes provenientes desde el TAS y mostrarlos al usuario y el *Command Manager* es quien recibe los pedidos del TAS para que el usuario introduzca información de configuración y envía al TAS las respuestas.

4.5.1. Respuesta a las solicitudes de TAS

Para tener una forma flexible de contemplar las distintas solicitudes de información o parámetros de configuración que pueden querer enviarse desde al TAS al usuario, se implementa el patrón de diseño RPC (Remote Procedure Call) utilizando el broker AMQP.

Cuando el TAS necesita enviar una solicitud que requiere intervención del usuario, primero crea una cola temporal en el broker para recibir la respuesta. Luego, al enviar la solicitud se indica en las propiedades del mensaje la routing key a la que se debe enviar la respuesta y un identificador único que permite asociar la respuesta con la solicitud enviada. Una vez enviada la solicitud el código que la realiza queda a la espera de recibir una respuesta en la cola creada para tal efecto.

La creación de este tipo de llamadas está implementada en la clase *RPCRequest* del módulo *user_interface.ui_reports* y el diagrama de la Fig. 4.19 muestra sus atributos y métodos.

Cuando desde el TSC se desea enviar una solicitud de configuración al usuario se crea un objeto de la clase *RPCRequest* indicando hacia donde enviar la solicitud (i.e. request_key). Luego se llama al método *RPCRequest.wait_response()* quedando bloqueado a la espera de la respuesta. Antes de publicar la solicitud en el exchange usando la request_key se crea una queue en donde se esperará a la respuesta. Por ejemplo, en caso de enviar una solicitud con la routing key *config.request* se espera que la respuesta venga dirigida a la routing key *config.reply*. La Fig. 4.20 ilustra este proceso en el que la *RPCRequest* creada se queda bloqueada a la espera de recibir una respuesta.

4.5.2. Formato de solicitudes y reportes

Esta sub-sección describe las clases que definen el formato para los reportes que el TAS envía a la Interfaz de Usuario, así como el formato para la solicitud de información. Ejemplos de información solicitada por el TAS al usuario para la

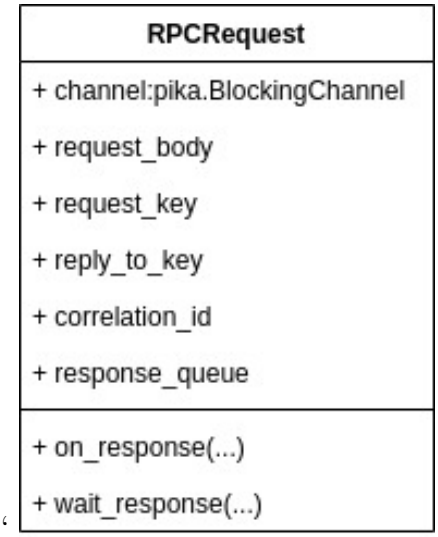

Figura 4.19: Clase que implementa las llamadas remotas desde el TAS hacia la Interfaz de Usuario.

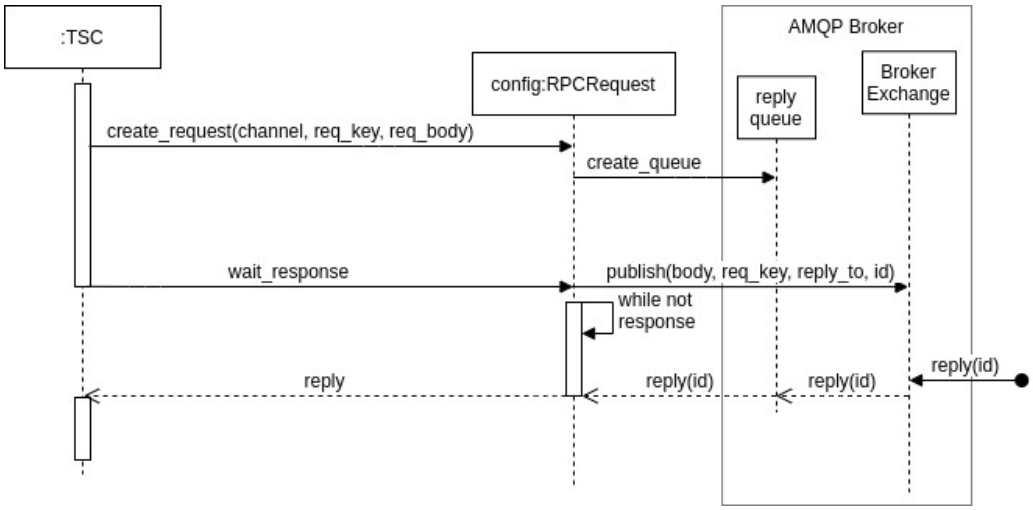

Figura 4.20: Creación de una solicitud desde el TSC y espera de la respuesta.

configuración de una sesión de test pueden ser los parámetros de registro del DUT así como solicitudes de enviar la señal de comienzo de tests una vez que el Agente se encuentra configurado y listo para comenzar.

El módulo *user_interface.ui_reports*, además de implementar la clase *RPCRequest* que maneja el envío de las solicitudes y recepción de respuestas, también define el formato de las solicitudes que se envía en el conjunto de clases de la Fig. 4.21.

Para cada solicitud de interacción por parte del usuario se crea un *InputFormBody* al que se le agregan campos con mensajes y requerimientos para el usuario. Estos campos heredan de la clase abstracta *InputField* y pueden ser: párrafos con

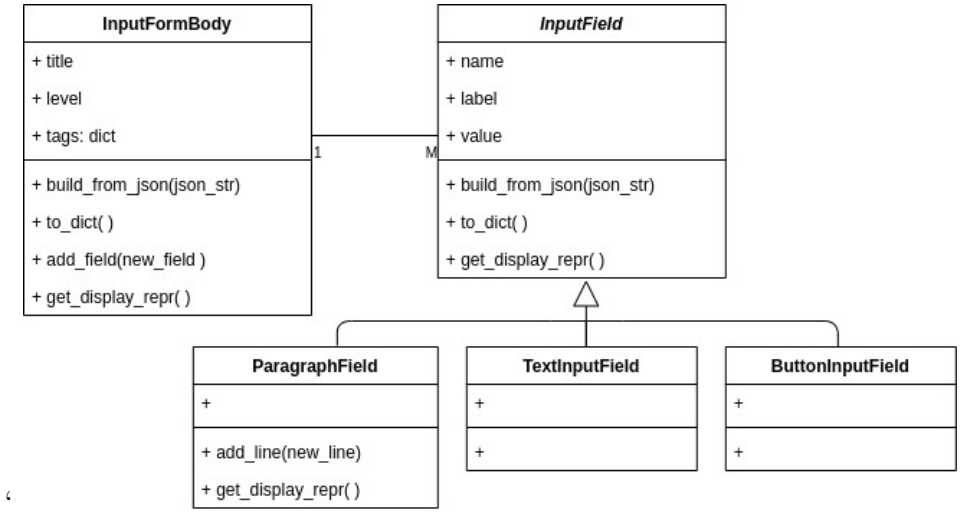

Figura 4.21: Clases que definen el formato para las solicitudes y notificaciones enviadas a la interfaz de usuario.

mensajes de información (*ParagraphField*), campos de texto para ser completados por el usuario (*TextInputField*) o botones que pueden insertarse en una interfaz gráfica para permitir al usuario por ejemplo dar comienzo a la sesión (*ButtonInputField*).

4.5.3. Comunicación de los servicios de la Interfaz de Usuario

En esta sección se explica la forma en la que se comunican los servicios que componen la interfaz de usuario usando el broker de mensajes RabbitMQ.

Una vez iniciada la interfaz de usuario el servicio de *Notification Display* crea queues y se suscribe a las siguientes routing keys en el Broker AMPQ, parametrizadas en el módulo *lorawan.parameters.message_broker*:

- {*routing_keys.users*}.*display*, para desplegar todos los mensajes dirigidos al usuario.
- {*routing_keys.configuration_request*}, permite mostrar cuando se recibe el pedido de configuración de la sesión de tests con el listado de los tests a ejecutar.
- {*routing_keys.users.request*}, para mostrar al usuario cuando se le ha enviado una solicitud.

Por otra parte el servicio *Command Manager* se encarga de procesar las solicitudes para que el usuario pueda interactuar, ya que el *Notification Display* solo despliega los mensajes y solicitudes sin procesar la respuesta del usuario. El *Command Manager* crea queues y se suscribe a las siguientes routing keys:

- {*routing_keys.configuration_request*}, al recibir el pedido de configuración espera que la misma sea introducida por el usuario y envía la respuesta utilizando la routing key (reply_to) incluida en el mensaje.

- {*routing_keys.users.request*}, para manejar el resto de solicitudes que se envían a la interfaz de usuario y enviar la respuesta.

La Fig. 4.22 muestra de forma esquemática la comunicación de la interfaz de usuario con el TAS usando el broker de mensajes, mostrando las queues que crea cada servicio y las respuestas enviadas por el *Command Manager*.

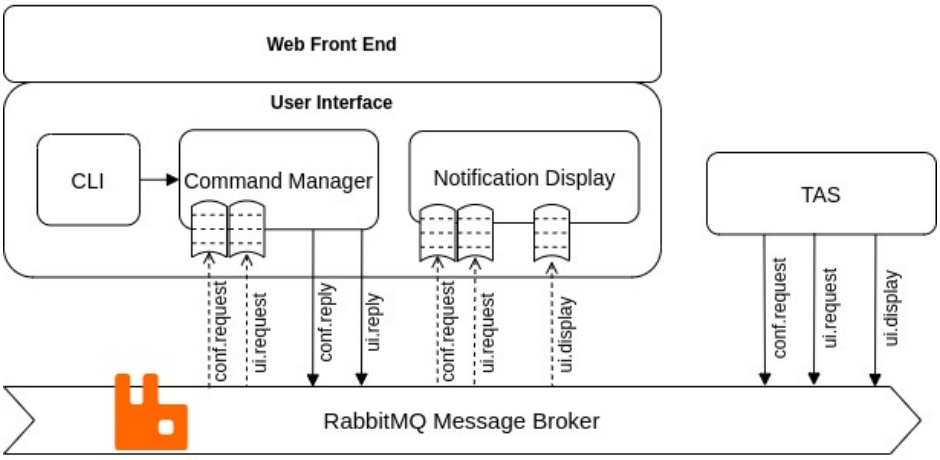

Figura 4.22: Servicios de la interfaz de usuario comunicándose con el TAS mediante AMQP a través del broker RabbitMQ.

4.5.4. Procesamiento de Comandos y Visualización de la Información

Tal como se explicó en la sección anterior, el servicio *Notification Displayer* se suscribe a los mensajes provenientes del TAS y tiene por lo tanto la información disponible para ser visualizada por el usuario. El *Notification Displayer* implementa un servidor y aplicación web desarrollada con Flask [7], y permite al usuario conectarse mediante un navegador web para interactuar con la sesión de test en el TAS.

Flask es un framework de Python que ofrece un conjunto de herramientas y librerías para la creación de aplicaciones web, es ampliamente utilizado y fácilmente extensible. Además, incluye un servidor web que si bien está orientado a testing y debugging, y no está recomendado para usarse en producción, es adecuado para la interfaz de la plataforma de tests diseñada ya que solamente se requiere conectar un único cliente.

La Fig. 4.23 muestra la relación y el flujo de mensajes entre los componentes de la interfaz de usuario.

[7] `https://flask.palletsprojects.com/en/1.1.x/`

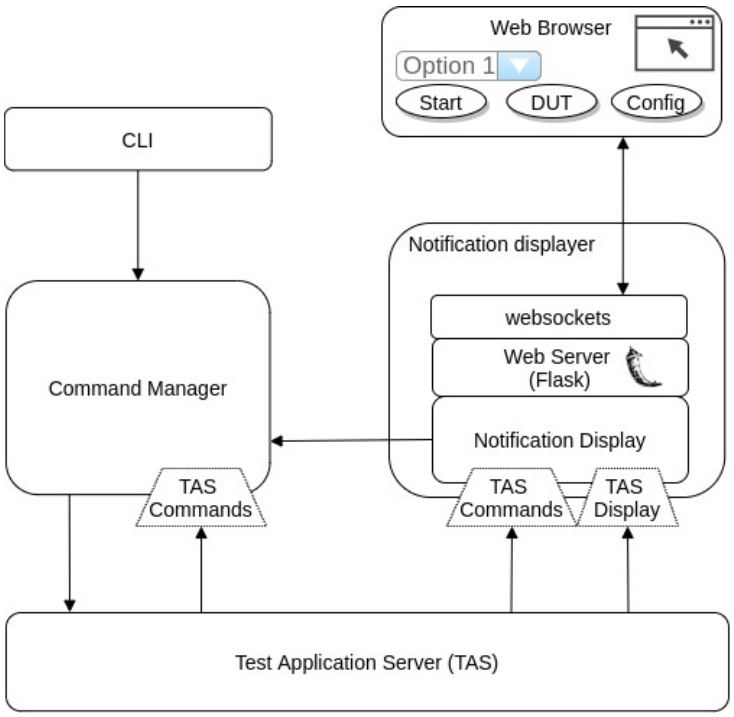

Figura 4.23: Implementación de la interfaz de usuario para visualizar información y enviar comandos al TAS.

La aplicación web mantiene una conexión con un navegador web del usuario utilizando Websockets [8] de forma de presentar en tiempo real los mensajes que se reciben desde el TAS. Cuando se detecta una solicitud del TAS para que el usuario envíe un comando como respuesta esta información es mostrada en un cuadro de Alertas. La interfaz web del *Notification Displayer* ofrece adicionalmente la posibilidad de generar comandos de:

- Configuración de la sesión con el listado de tests a ejecutarse.
- Parámetros de identificación de DUT y clave raíz *appKey*.
- Inicio de la sesión de tests.

En la Fig. 4.24 se puede ver una captura de pantalla de la interfaz web implementada.

Ya sea cuando el usuario envía comandos por la interfaz web del *Notification Displayer* o a través de la CLI, estos son dirigidos hacia el servicio de *Command Manager*, que se encarga de procesarlos y enviarlos al TAS. Estos comandos pueden ser: la configuración de los tests que deben ejecutarse, las credenciales LoRaWAN para el registro del DUT y la señal de inicio de sesión de test. Adicionalmente, este componente espera, procesa y envía la respuesta a cualquier solicitud de acuerdo a lo explicado en la Subsección 4.5.1.

[8] `https://python-socketio.readthedocs.io/en/latest/intro.html#what-is-socket-io`

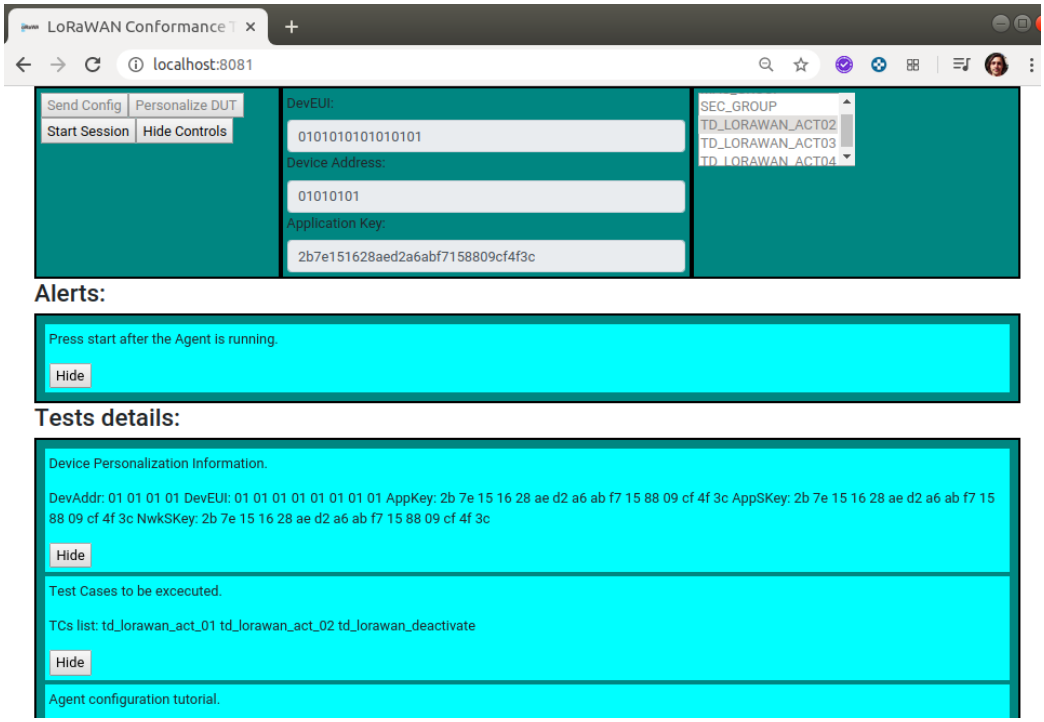

Figura 4.24: Captura de pantalla de la Interfaz Web implementada para interactuar con el TAS.

4.6. Despliegue de los servicios

Para ejecutar una sesión de tests en un entorno local, tal como se describe en la Sección 3.3.1 del Capítulo 3, se deben iniciar, configurar y conectar los contenedores Docker con los siguientes servicios:

- message-broker: RabbitMQ.
- test-application-server: servicio principal de la plataforma de pruebas.
- command-manager: componente de la interfaz de usuario encargado de procesar los comandos.
- notification-displayer: aplicación web de la interfaz de usuario.
- agent: conector entre el DUT y LoRa gateway proporcionados por el usuario y el TAS.
- cli: contenedor con herramientas de línea de comandos incluidas dentro de la interfaz de usuario.

Se utilizó *Docker Compose* [9], herramienta que permite simplificar la gestión de contenedores Docker. Con esta herramienta se puede desplegar una aplicación

[9] `https://docs.docker.com/compose/`

compuesta por múltiples servicios a partir de un fichero de configuración que especifica la imagen con la que se debe crear cada contenedor, los puertos que se desea tener abiertos hacia el exterior, variables de entorno y comandos que se deben ejecutar.

En el Apéndice B se encuentra disponible el fichero de configuración *docker-compose.yml* usado por *Docker Compose*, junto a los ficheros *Dockerfile* de los servicios definidos y un Makefile que se proporciona para facilitar el despliegue. Además se incluye una guía rápida para desplegar la plataforma, con los requerimientos y los pasos que deben seguirse.

4.7. Resumen y conclusión del capítulo

En este capítulo se explicó cómo fue implementado el diseño mostrado en el Capítulo 3. Se mostraró detalladamente cómo se utilizó RabbitMQ como broker de mensajes aprovechando las características de este servicio para implementar una arquitectura de RPC para gestionar el envío de comandos. La implementación del TAS fue detallada mostrando cómo el sistema de manejo de excepciones de Python es aprovechado para reportar los errores detectados e incluirlos en el informe que se envía al usuario. La interacción con el usuario de la plataforma se maneja con los servicios que componen la interfaz gráfica, herramienta que incluye una aplicación web y una interfaz por línea de comandos.

La parte restante del documento se enfocará en describir las pruebas realizadas a la plataforma y mostrar un ejemplo de integración de la misma en un entorno de tests.

Capítulo 5

Resultados Obtenidos

En este capítulo se muestran los resultados obtenidos al evaluar la plataforma de pruebas de conformidad LoRaWAN. Se detallan los dispositivos utilizados y las versiones del firmware con la implementación de LoRaWAN sometida a pruebas, explicando algunas modificaciones introducidas para inyectar errores específicos y así comprobar que estos son detectados de forma satisfactoria.

El capítulo comienza introduciendo algunas herramientas auxiliares que fueron desarrolladas para facilitar el análisis de los mensajes LoRaWAN intercambiados entre el DUT y el TAS. Luego la Sección 5.2 describe la información incluida en los reportes finales de las sesiones de test. El capítulo continúa especificando en la Sección 5.3 los dispositivos utilizados para probar la plataforma, explicando los resultados obtenidos al ejecutar los tests implementados y en la Sección 5.5 se analiza cómo los errores inyectados se ven reflejados en los resultados de los tests. Finalmente en la Sección 5.5 se describe la integración de la plataforma de conformidad LoRaWAN con el entorno de tests F-Interop.

5.1. Herramientas para el Análisis de Resultados

En algunas situaciones, como cuando se detecta un error en el formato de un mensaje, puede ser suficiente con ver el mensaje recibido para entender por qué el test ha fallado. Sin embargo, en otros casos se requiere ver la secuencia de mensajes anteriores para poder identificar dónde se ha producido el error. Para estos últimos casos, y en general para poder analizar en detalle la secuencia de mensajes, se proporciona un nuevo componente encargado de inspeccionar el tráfico entre el DUT y el TAS. Este componente auxiliar llamado *Message Inspector* recibe todos los mensajes de downlink y uplink, los procesa, desencripta y genera un log con su contenido junto con la información de timestamp, frecuencia del canal utilizado y data rate.

La Fig. 5.1 muestra cómo el *Message Inspector* interactúa con los componentes de la interfaz de usuario detallados en la Subsección 4.5.4 y el TAS. Debido a la arquitectura de comunicación seleccionada en la Subsección 4.2.2 los componentes existentes no se ven afectados y solamente hace falta registrar nuevas *queues* para

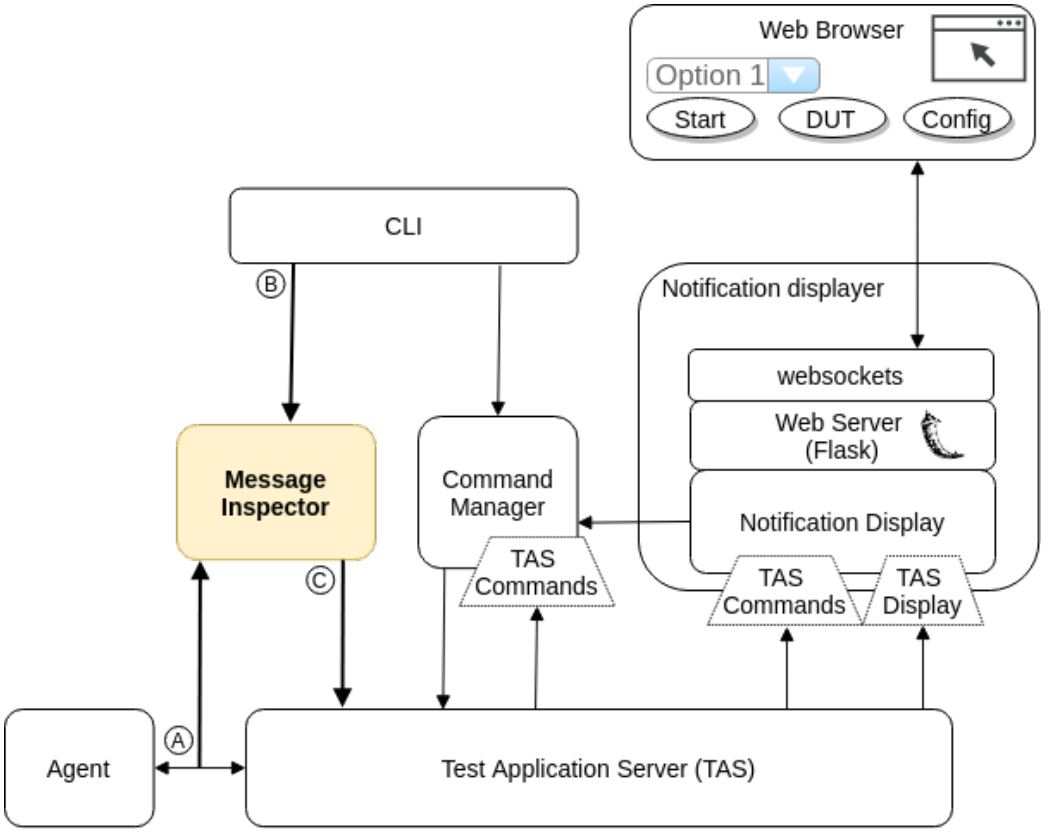

Figura 5.1: Message Inspector preparado para recibir copia de los mensajes intercambiados para analizarlos y enviando mensajes simulados al TAS.

el *Message Inspector* y asociarlas a la *routing key* deseada.

Al iniciar el *Message Inspector* se le configuran las credenciales usadas por el DUT y de esta forma cuenta con toda la información para procesar los intercambios de mensajes de JOIN [1] e ir actualizando las claves de sesión LoRaWAN.

Un ejemplo de cómo la información de este componente puede complementar al reporte final y a los logs del Agente se muestra en la Fig. 5.2. Para hacer un seguimiento de los mensajes y encontrar fácilmente la correspondencia entre estos en diferentes logs se puede utilizar el timestamp incluido por el Gateway LoRa. El timestamp es un entero que representa el tiempo interno del Gateway, sus unidades representan milisegundos, y más que su valor absoluto lo que nos interesará son las diferencias de tiempo entre diferentes mensajes. Basados en el timestamp del mensaje que genera el error en el test de la Fig. 5.5 (i.e. 800658804) se puede localizar el mensaje anterior y ver que correspondía a un PING y es por eso que se esperaba la respuesta PONG.

Finalmente, al observar los logs del Agente en la Fig. 5.3 se puede comprobar que el gateway en este caso reportó un error al recibir el mensaje desde el Agente y por lo tanto este no fue agendado para enviarse al DUT. La Subsección 4.4.1

[1] Este proceso se explica detalladamente en la Sección 2.4.3.

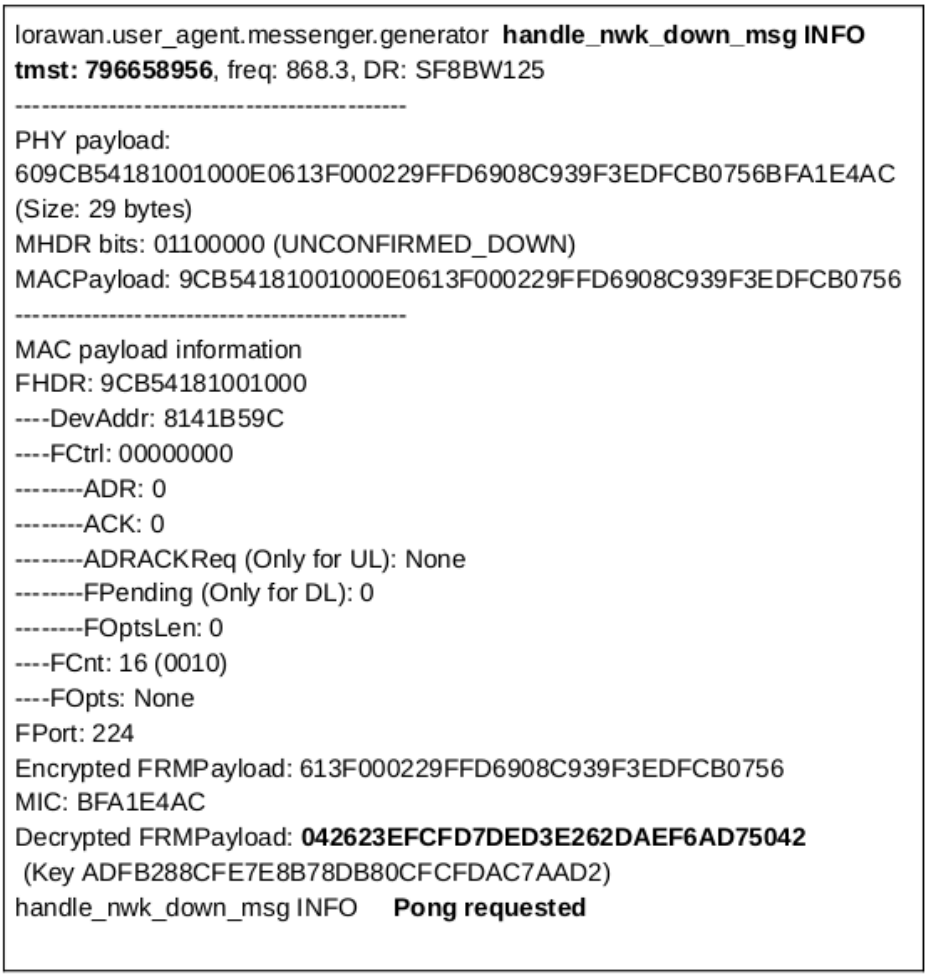

Figura 5.2: Información contenida en los logs del Message Inspector mostrando el mensaje de downlink enviado al DUT justo antes de recibirse en el TAS el mensaje que genera el error en el test.

explica el intercambio de mensajes entre el Agente y el Gateway LoRa basado en el protocolo Semtech Packet Forwarder.

Adicionalmente al análisis del intercambio de mensajes, el *Message Inspector* es capaz de recibir solicitudes generadas por la interfaz de línea de comandos para generar e insertar mensajes LoRaWAN, manteniendo una instancia de un DUT simulado que imita su funcionamiento, de ahí el término *mock* en el nombre de la función. Este *mock*, instancia simulada del DUT, debe generar los mensajes LoRaWAN encriptando el payload y mantener la información de sesión actualizada al momento de intercambiar mensajes de Join. En la Fig. 5.1 se muestra cómo además del análisis de los mensajes, representado por la flecha que va hacia el Message Inspector en el punto A, también se reciben solicitudes desde la CLI en el punto B y se envían mensajes LoRaWAN hacia el TAS en el punto C.

A continuación se listan los comandos disponibles desde la *CLI* que permiten

Figura 5.3: Error reportado por el Agent, con el mensaje del Gateway indicando que el mensaje de Downlink no fue enviado al DUT.

simular mensajes LoRaWAN, pudiendo de esta forma hacer pruebas básicas con el TAS sin necesidad de conectar un DUT y un gateway LoRa:

- *configure_device_mock*: permite especificar los parámetros usados para los mensajes simulados. Adicionalmente, permite reiniciar la sesión del dispositivo indicando que debe usar las claves ABP inicialmente asignadas.
- *show_info*: muestra la configuración actual del simulador de mensajes, con las claves de sesión que actualmente tiene configuradas en base al intercambio de mensajes procesado.
- *send*: envía un mensaje LoRaWAN pasando como parámetro con el FRM-Payload especificado y los parámetros configurados para el dispositivo simulado.
- *send_join*: envía un mensaje de Join Request solicitando al TAS una nueva sesión.
- *send_actok*: genera un mensaje de TAOK con el contador de downlink.
- *send_pong*: envía un mensaje PONG con la respuesta al último mensaje PING detectado.

En la siguiente sección se describe la información proporcionada al usuario en los reportes finales que muestra la interfaz web.

5.2. Información del Reporte Final de cada Sesión

De acuerdo a lo explicado en la Sección 4.5 la interfaz de usuario web despliega un reporte al finalizar la sesión en donde se resumen los resultados de las pruebas seleccionadas. La Fig. 5.4 muestra un ejemplo en donde se reporta un error en el test *td_lorawan_sec_01*, que en este caso ha fallado por haber recibido un mensaje que no era el esperado en ese paso del test. Entre la información con los motivos del error que se adjunta en este informe se encuentra el último mensaje recibido;

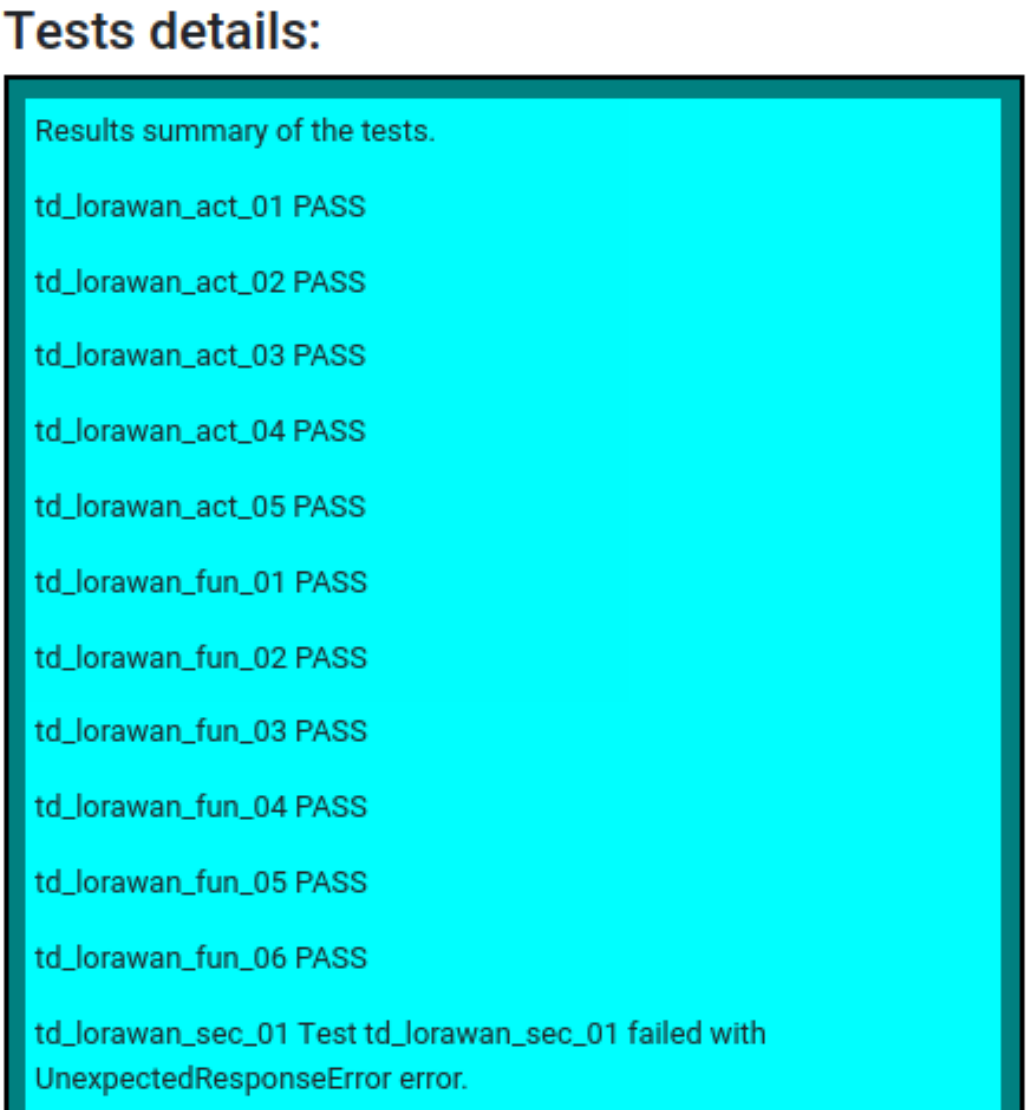

Figura 5.4: Ejemplo de un reporte final de una sesión de test.

esto se muestra en la Fig. 5.5 donde se aprecia la metadata referente al instante de recepción del mensaje, la frecuencia utilizada en el canal de radio y el Data Rate. En la Fig. 5.5 se puede ver que el mensaje recibido se decodifica y analiza mostrando cada uno de sus campos, además de desencriptar el payload y mostrar la clave de sesión usada para esto.

La siguiente sección analiza los resultados obtenidos al ejecutar los tests de conformidad en un dispositivo con una implementación de LoRaWAN ya certificada.

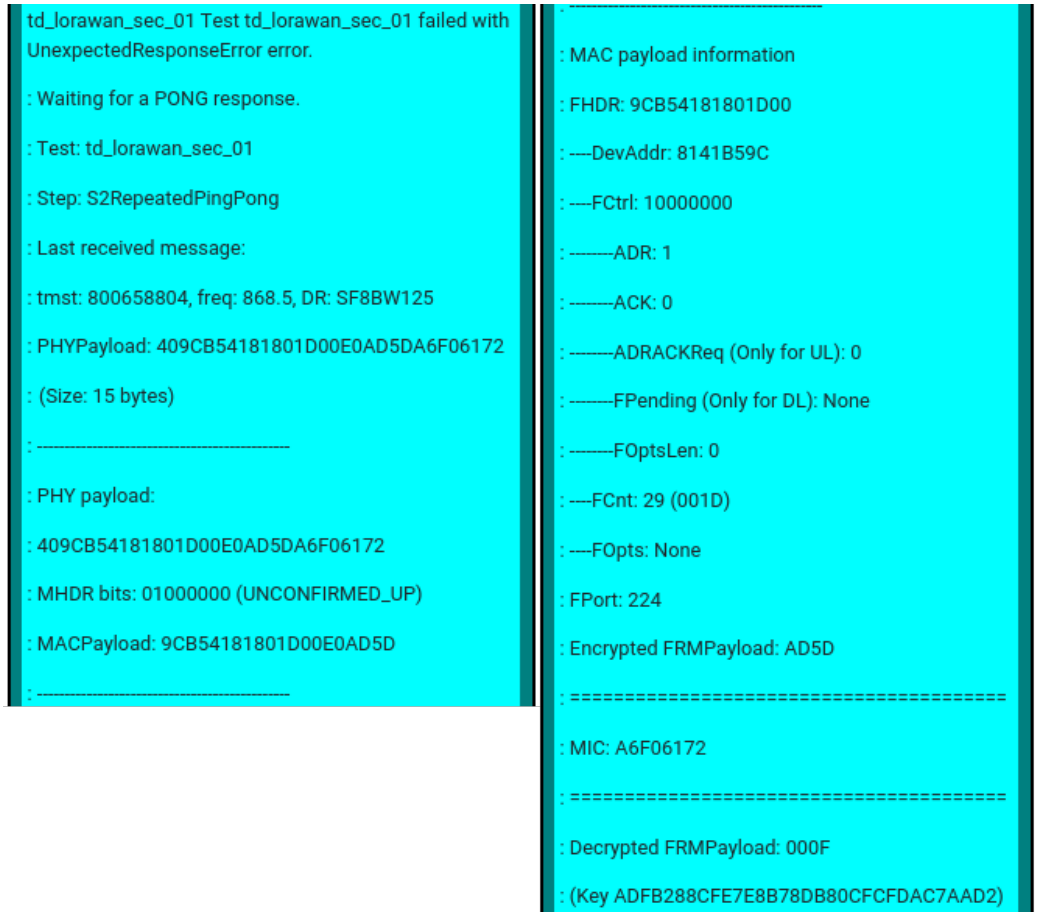

Figura 5.5: Error detectado en el test *td_lorawan_sec_01* debido a recibir un mensaje de TAOK cuando se esperaba un PONG.

5.3. Ejecución de Tests en un Dispositivo Certificado

Para probar la plataforma de tests implementada se utilizó el Discovery Kit B-L072Z-LRWAN1 de ST Microelectronics [22] y un gateway LoRa Kona Micro de Tektelic Communications [12], ambos mostrados en la Fig. 5.6. El Discovery Kit B-L072Z-LRWAN1 es una plataforma de desarrollo que soporta comunicación LoRa mediante la incorporación del módulo de Murata CMWX1ZZABZ-091 [19], que integra un microcontrolador y un módulo de radio LoRa.

La implementación de LoRaWAN utilizada fue la del paquete de librerías y ejemplos I-CUBE-LRWAN Expansion Package [21] en la versión 1.1.5, que soporta (entre otras) la región EU868 de LoRaWAN y además incluye la aplicación de test preparada para el proceso de certificación. De esta forma contamos con un módulo que ya ha pasado las pruebas de certificación [10] y una implementación del protocolo de aplicación de test que se describió en la Sección 3.4.

El dispositivo se configura con los siguientes parámetros:

- Inicia con tipo de activación ABP, por lo que al encenderlo comienza a enviar

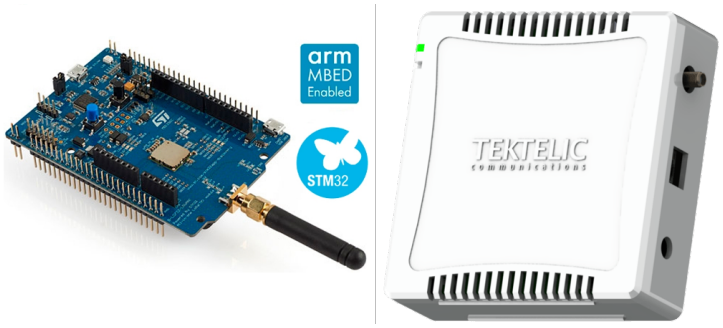

Figura 5.6: ST Discovery Kit B-L072Z y Gateway Tektelic Kona Micro, dispositivos utilizados para probar la plataforma de tests de conformidad LoRaWAN.

datos utilizando las claves de sesión preconfiguradas.

- DevAddr = 01010101
- AppSKey = FF7E151628AED2A6ABF7158809CF4F3C: clave de sesión de aplicación usada para encriptar los datos.
- NwkSKey = 007E151628AED2A6ABF7158809CF4F3C: clave de sesión de red usada para calcular el MIC y encriptar comandos MAC.
- AppKey = 2B7E151628AED2A6ABF7158809CF4F3C: clave raíz que se utilizará al realizar el proceso de Join Request durante una activación de tipo OTAA.
- DevEUI = 0101010101010101: identificador del dispositivo que será usado al iniciar un proceso de Join Request.
- AppEUI = 0101010101010101: identificador de la aplicación correspondiente al TAS que será incluido en los mensajes de Join Request.

A continuación se analizan los resultados observados al ejecutar cada uno de los tests, el reporte completo junto con los logs del Agente y Message Inspector están disponibles en el repositorio del código en github [2]. Como era esperable tratándose de una implementación ya probada, todos los tests pasan satisfactoriamente.

5.3.1. Tests de Activación

La descripción detallada de todos los test de activación implementados se encuentra en la Subsección A.1.1 del Apéndice. Al observar la información desplegada por la interfaz web cuando se ejecuta el primer test *td_lorawan_act_01* vemos que tal como se esperaba el primer paquete recibido desde el DUT contiene un mensaje de datos encriptado con la AppSKey preconfigurada, calcula el MIC con la NwkSKey esperada y tiene la DevAddr asignada. La Fig. 5.7 muestra la información más

[2] `https://github.com/pablomodernell/lorawan_conformance_testing/tree/master/reports_logs_examples`

Figura 5.7: Información mostrada por la interfaz web al recibir el primer mensaje desde el DUT durante la ejecución del test *td_lorawan_act_01*.

relevante mostrada al finalizar el primer paso de este test *S1DataToActivate*, mientras que la Fig. 5.8 muestra cómo con el *Message Inspector* se puede comprobar que el TAS agenda el mensaje de downlink con los parámetros esperados.

El siguiente mensaje enviado por el DUT es un TAOK, comprobando que el mensaje de downlink mostrado en la Fig. 5.8 fue recibido correctamente.

Otro punto a destacar en el análisis de los mensajes intercambiados durante los tests de activación es cuando el DUT solicita iniciar una nueva sesión con el mecanismo de OTAA. Tal como se explica en la descripción de los tests de la Subsección A.1.1 este evento ocurre debido a una solicitud del TAS usando un mensaje de Trigger Join según lo especificado en protocolo de test explicado en la Sección 3.4. La Fig. 5.9 muestra la información disponible en la interfaz web cuando el TAS procesa un Join Request y la Fig. 5.10 permite observar cómo el mensaje de Join Accept es agendado para su envío con $JOIN\_ACCEPT\_DELAY1$=5, 5 segundos después, de acuerdo a lo esperado.

Al analizar el mensaje siguiente enviado por el DUT en la Fig. 5.11 luego de enviado el mensaje de Join Request vemos que se utilizan correctamente las nuevas claves para encriptar los datos y para generar el MIC, y que la nueva dirección usada es la asignada en el proceso de OTAA.

Luego de ejecutados todos los tests de activación se pudo comprobar que el intercambio de mensajes es el esperado, pudiendo realizarse las activaciones en ambas ventanas de downlink y siendo posible configurar nuevas frecuencias utilizando el mensaje de Join Accept. La Fig. 5.12 muestra una captura de la interfaz web al completarse el cuarto paso del test *td_lorawan_act_04*, informando que luego

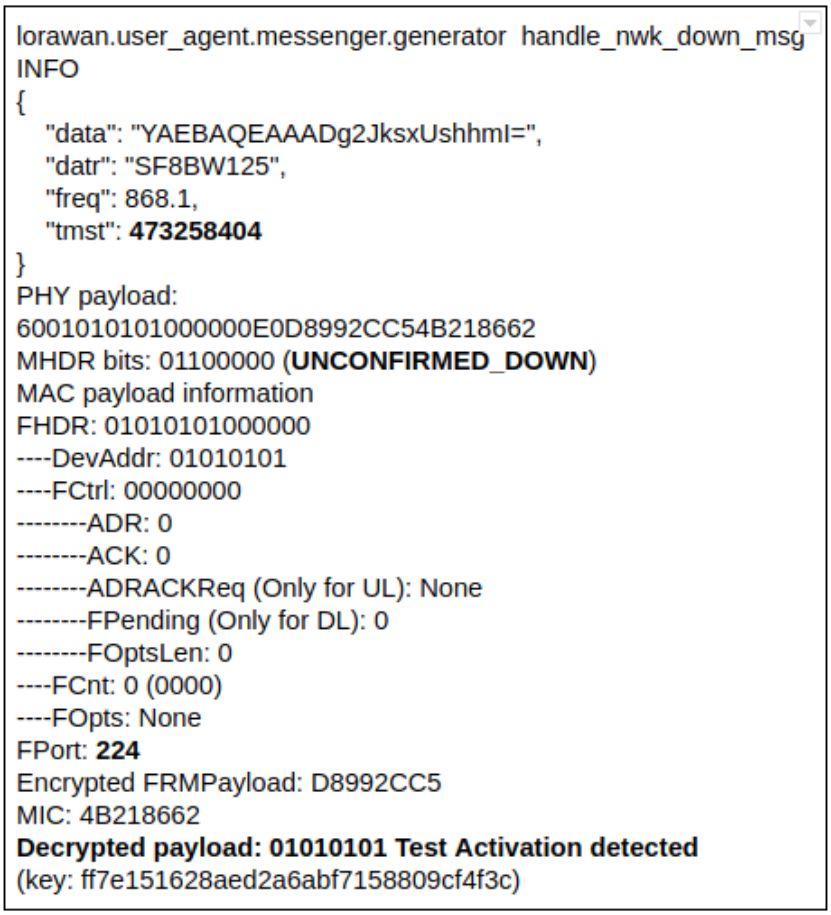

Figura 5.8: Información contenida en los logs del Message Inspector mostrando el mensaje de downlink agendado con la activación del modo test en el primer paso del test *td_lorawan_act_01*.

de recibir 17 mensajes se logró comprobar que el DUT usa todas las frecuencias configuradas.

5.3.2. Tests de Funcionalidades Básicas y Tiempos

Entre los aspectos a destacar de los resultados observados al ejecutar los test del grupo de funcionalidades básicas y tiempos (*FUN*), se pudo comprobar que el DUT cumple con la tolerancia esperada ante errores de tiempo de recepción para los mensajes de Downlink. En la Fig. 5.13 se observa el instante de recepción de un mensaje de uplink al completarse el primer paso de test *td_lorawan_fun_01* (i.e. 69**0**661**3**2), mientras que en la Fig. 5.14 se puede ver cómo el mensaje de downlink siguiente es agendado 20 microsegundos más tarde del tiempo de inicio de la ventana RX1 (i.e. recordando que la ventana se debería iniciar 1 segundo después: 69**1**661**9**2 en vez de 69**1**661**3**2, en negrita el millón de microsegundos, y las decenas de microsegundos mostrando el error).

Además de verificar la tolerancia a los adelantos y retrasos temporales respecto

Figura 5.9: Información desplegada por la interfaz web al momento de procesar un mensaje de Join Request asignando una nueva sesión.

al inicio de ambas ventanas de Downlink, los test del grupo *FUN* comprobaron satisfactoriamente que el número de secuencia se incrementa en los mensajes de uplink, que el DUT ignora un mensaje si este se envía con un FCnt menor al último utilizado y que los mensajes con confirmación son manejados correctamente.

5.3.3. Tests de Mecanismos de Seguridad

Los tests del grupo *SEC* para los mecanismos de seguridad verificaron las características la confidencialidad e integridad de mensajes de LoRaWAN. Sin embargo algunas de estas funcionalidades ya habían sido probadas en los tests anteriores, ya que si el DUT no implementara correctamente la encriptación el intercambio de mensajes PING-PONG habría fallado. Los tests *td_lorawan_sec_01* y *td_lorawan_sec_02* prueban específicamente la encriptación en el DUT, enviando una secuencia de varios mensajes con distintos tamaños. Adicionalmente, se comprobó que el DUT ignora un mensaje enviado con el MIC incorrecto tal como indica la especificación de LoRaWAN.

5.3.4. Tests de Intercambio de Comandos MAC

Utilizando los tests del grupo *MAC* se probó satisfactoriamente el procesamiento de los comandos MAC en el DUT. En la Fig. 5.15 se muestra de qué forma la interfaz web informa de que un comando MAC fue enviado hacia el DUT, indicando si el mismo fue incluido en el encabezado de la trama (FHDR) o si

Figura 5.10: Información mostrada en el log del Message Inspector donde se ve el timestamp que indica el momento en el que se agenda el mensaje de downlink con el Join Accept.

fue enviado en el payload y por lo tanto encriptado usando la clave de sesión de red (i.e. *NwkSKey*). La respuesta enviada por el DUT a este comando se recibió correctamente y puede verse en la Fig. 5.16.

Entre las verificaciones realizadas se comprobó que el DUT ignora un mensaje con comandos MAC presentes simultáneamente en el encabezado de trama (i.e. *FHDR*) y en el payload de acuerdo a lo establecido por la especificación de LoRaWAN. También se probó eliminar los canales por defecto de la región EU, verificando que el DUT responde a este comando rechazando el cambio. Todas las pruebas realizadas con intercambios de comandos MAC tuvieron el resultado esperado, los reportes completos se encuentran disponibles en el repositorio github junto al código de la plataforma [3].

La siguiente sección describe algunas modificaciones que fueron realizadas al firmware del DUT para inyectar errores, mostrando cómo estos errores en la implementación de LoRaWAN o en la aplicación de test se ven reflejados en los

[3] `https://github.com/pablomodernell/lorawan_conformance_testing/tree/master/reports_logs_examples`

Figura 5.11: Mensaje de datos enviado por el DUT luego de actualizada la sesión, utilizando ahora las nuevas claves y DevAddr asignadas.

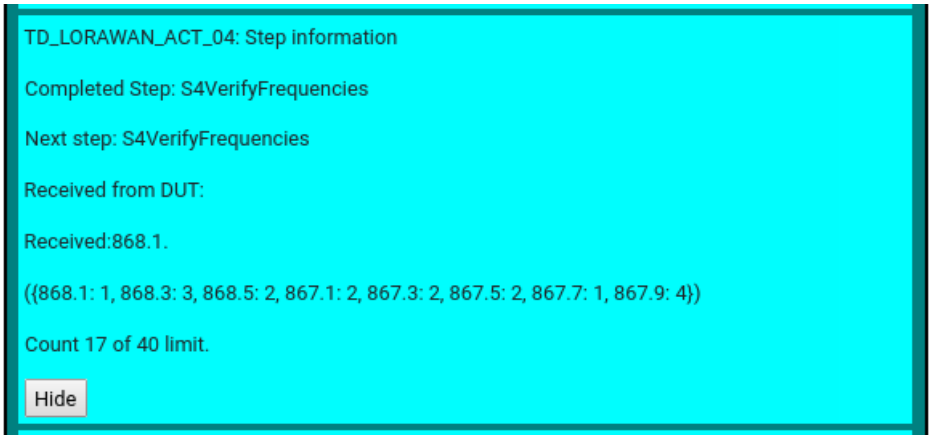

Figura 5.12: Información mostrada por la interfaz web al finalizar el test de activación *td_lorawan_act_04*, indicando que al menos un mensaje se ha recibido en cada una de las frecuencias configuradas para el DUT.

resultados de los tests.

5.4. Inyección de Errores

Las pruebas realizadas en la sección anterior permitieron demostrar el funcionamiento de la plataforma de conformidad LoRaWAN ante un dispositivo que

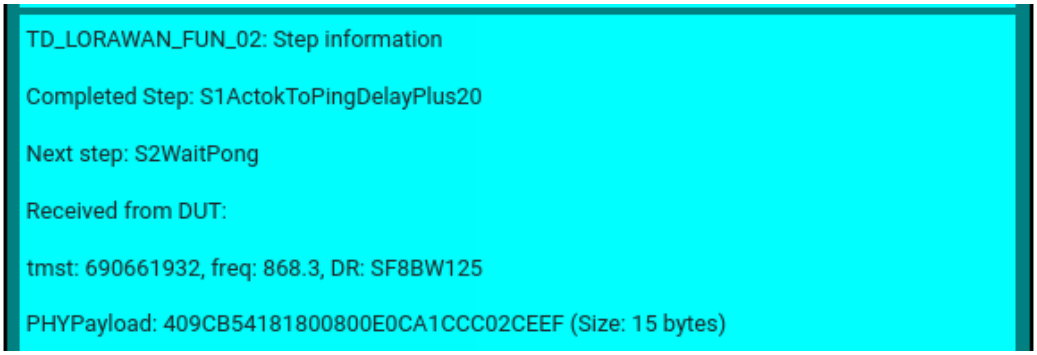

Figura 5.13: Recepción de un mensaje de Uplink que indica el tiempo en el que se abrirán las ventanas de Downlink.

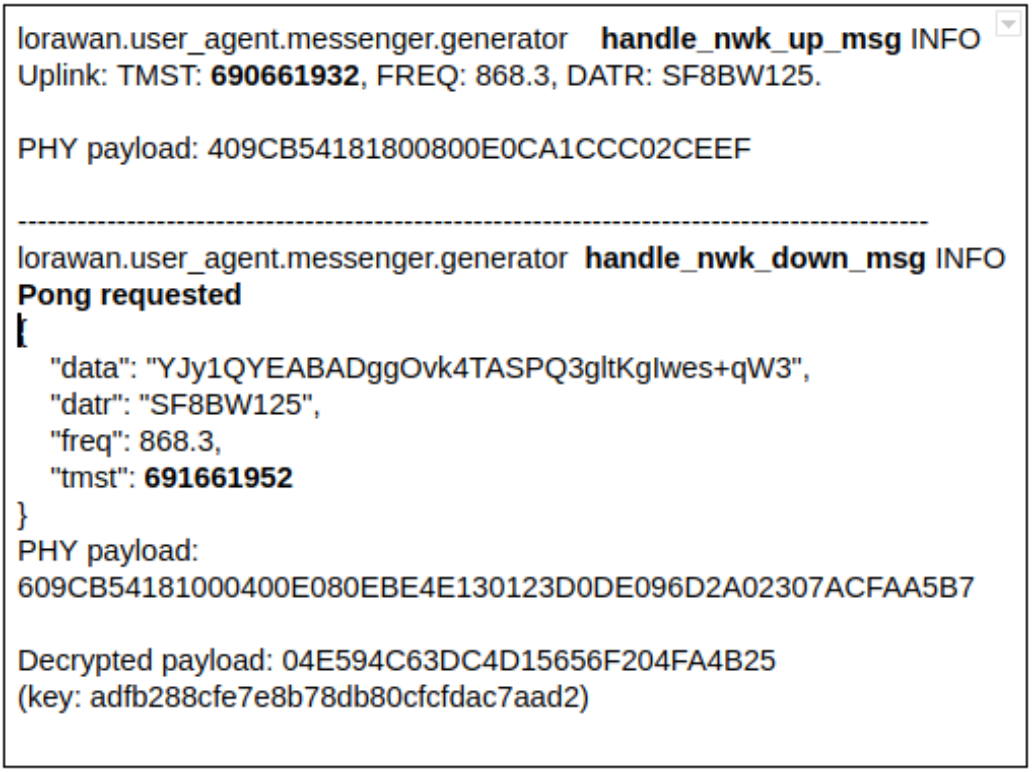

Figura 5.14: Información mostrada por el Message Inspector donde se observa que el mensaje de downlink está agendado 20 microsegundos más tarde del inicio de la ventana de Downlink RX1.

implementa correctamente todas las características especificadas por el protocolo. En esta sección se mostrará los resultados obtenidos cuando el DUT no se comporta de acuerdo a lo requerido por LoRaWAN y el protocolo de aplicación de tests.

En cada una de las siguientes subsecciones se describe un error introducido y se muestra cuál es la información proporcionada para poder identificarlo. Inicial-

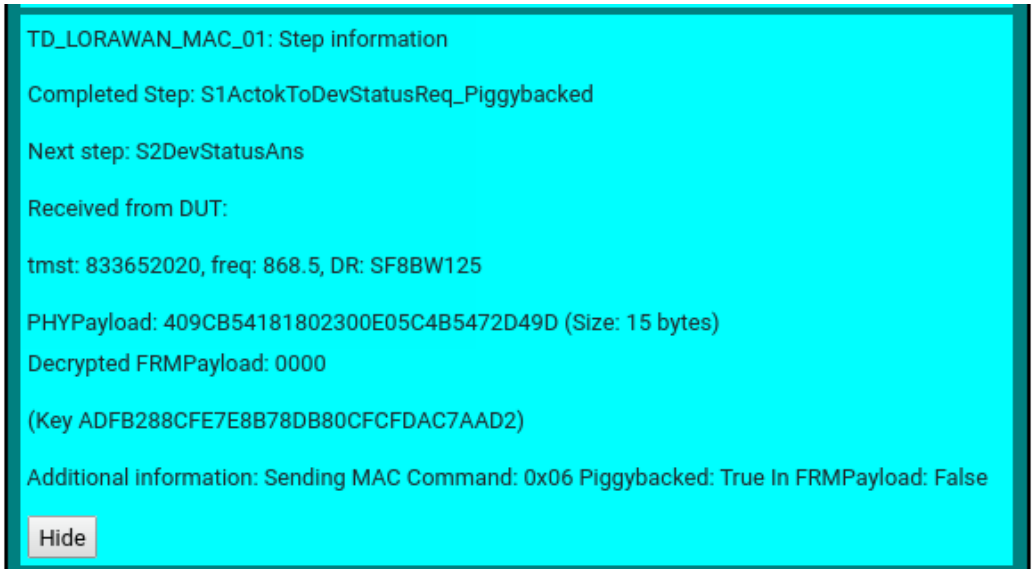

Figura 5.15: Información mostrada por la interfaz web al recibir un mensaje de Uplink y agendar uno de Downlink con un comando MAC en el FHDR.

mente se describen dos modificaciones que afectan la aplicación de test, y no a la implementación de LoRaWAN sometida a pruebas, ya que errores al nivel de esta aplicación también afectarían el resultado de los tests. Todas las modificaciones se introducen de forma independiente y aislada a las demás, volviendo a restaurar el código del DUT al estado original antes de introducir un nuevo cambio.

5.4.1. Errores en el Protocolo de Aplicación de Tests

En la primera prueba realizada se modificó la implementación del protocolo de aplicación de tests detallado en la Sección 3.4 haciendo que las respuestas a un mensaje de PING sumen 2 en vez de 1 al valor de cada byte. De esta forma, la respuesta a un mensaje de PING con contenido 0401aa22 será 0403ac24 en vez de 0402ab23.

Esta modificación sólo afecta los intercambios de mensajes PING-PONG y no afecta el resto de la implementación de LoRaWAN. En la Fig. 5.17 se ve la información mostrada al usuario cuando un test falla porque el mensaje PONG recibido como respuesta no es el esperado.

El siguiente error introducido fue la modificación del contador de mensajes de downlink incluido en los TAOK de la aplicación de test, haciendo que este no se incremente y se envíe siempre el valor 0 independientemente de la cantidad de mensajes de downlink recibidos. Un ejemplo de la forma en la que esta situación es reportada se muestra en la Fig. 5.18: el primer paso del test *td_lorawan_fun_01* falla al comprobar el mensaje de TAOK, debido a que el test anterior (i.e. *td_lorawan_act_05*) envió dos mensajes de PING antes de finalizar y sin embargo el Downlink Counter del TAOK recibido es 0.

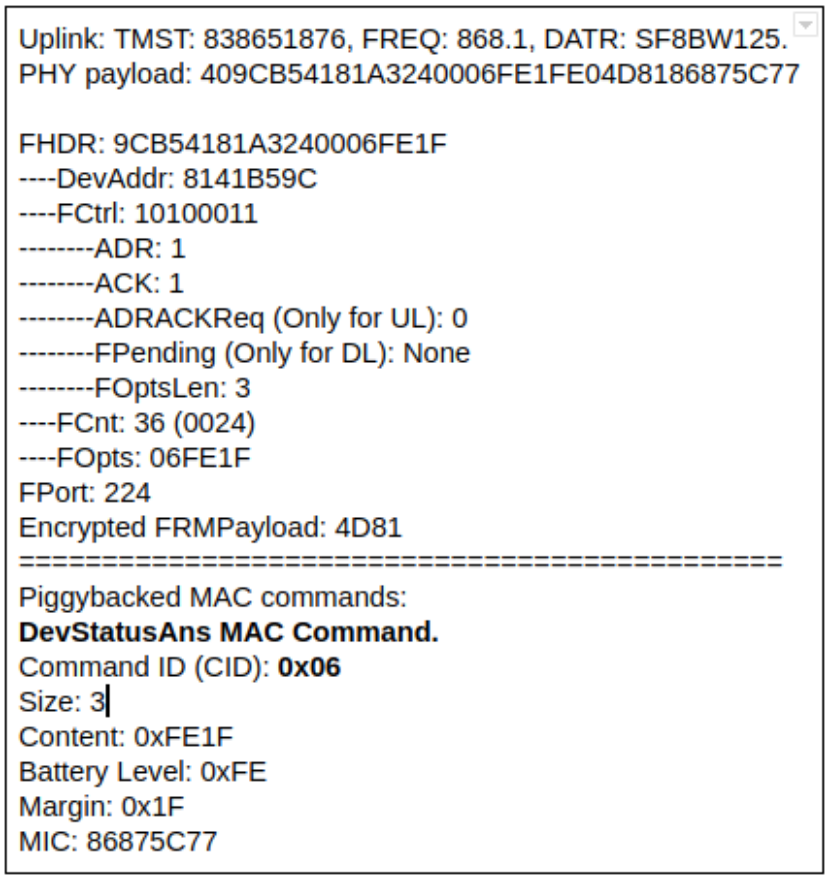

Figura 5.16: Mensaje de Uplink recibido con la respuesta al comando MAC enviado en el paso 2 del test *td_lorawan_mac_01*.

5.4.2. Procesamiento de Comandos MAC

Tal como fue explicado en la Subsección 2.5.3, uno de los requerimientos de LoRaWAN es que un mensaje sea descartado si contiene comandos MAC simultáneamente en el FHDR y en el FRMPayload (usando el FPort 0). Para demostrar cómo una falla de este tipo es reportada por el test *td_lorawan_mac_02* se modificó el firmware del DUT para que se procesen todos los mensajes a pesar de tener FOptsLen mayor a cero y FPort igual a cero. La Fig. 5.19 muestra el reporte donde se indica que se ha detectado esta situación, mientras que en la Fig. 5.20 se ve el mensaje enviado según lo mostrado por el Message Inspector.

5.4.3. LoRaWAN Uplink Frame Counter

Con el objetivo de mostrar cómo se detectan errores en el manejo de los contadores de tramas (i.e. FCnt, Frame Counter) se modificó la implementación del firmware del DUT para que no se incremente el Uplink Frame Counter. El test *td_lorawan_fun_03*, diseñado para probar que este contador se incrementa correctamente, indica el error detectado de la forma mostrada en la Fig. 5.21.

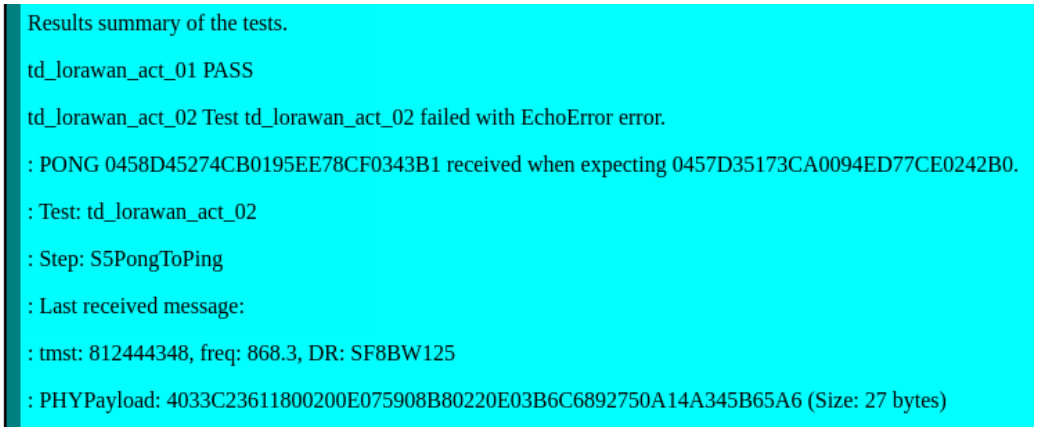

Figura 5.17: Reporte final indicando el fallo de un test debido al error inyectado en la generación de las respuestas PONG.

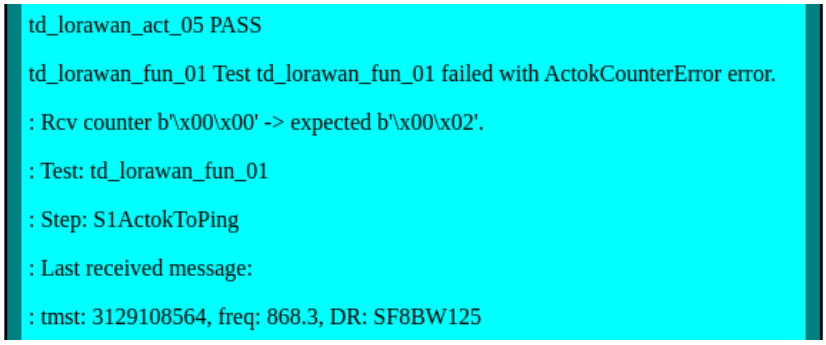

Figura 5.18: Error detectado al comprobar que el mensaje del TAOK continúa estando en 0 a pesar de haberse enviado dos mensajes de downlink hacia el DUT.

5.4.4. Gestión de Canales de Radio

LoRaWAN para la región europea especifica tres canales por defecto que deben ser siempre soportados y el DUT no debe dejar de utilizarlos aunque el servidor de red lo solicite a través de comandos MAC. Se modificó el firmware del DUT para que evite esta verificación y acepte un pedido de remover un canal aunque sea uno de los 3 canales por defecto (i.e 868.1, 868.3 y 868.5 MHz). La Fig. 5.22 muestra cómo al recibir un mensaje desde el DUT el TAS agenda un mensaje de Downlink que contiene dos comandos MAC deshabilitando los dos primeros canales por defecto. La Fig. 5.23 muestra cómo en el paso siguiente el test falla debido a que se detecta que el DUT aceptó el cambio.

5.4.5. Generación de las Claves de Sesión

Para simular un escenario en el que el DUT implemente mal la generación de las claves de sesión (i.e. NwkSKey y AppSKey) se modificó el firmware para que al procesar los mensajes de Join Accept genere la AppSKey exactamente igual a la NwkSKey. Al iniciar una sesión de tests con esta configuración se puede ver que

una vez renovada la sesión el dispositivo ya no es capaz de decodificar los mensajes recibidos desde el TAS y por lo tanto no inicia nunca la sesión de tests.

En este caso el error reportado por la interfaz web indica que se volvió a recibir un mensaje de datos en vez de un TAOK confirmando la activación del modo test. Para poder identificar el error es necesario conocer cuál es el mensaje de datos que el DUT envía, ver que el TAS desencripta y obtiene un mensaje diferente, y así comprobar que el problema está relacionado con la clave AppSKey utilizada por el DUT.

El error reportado cuando falla el test *td_lorawan_act_03* se muestra en la Fig. 5.24. A partir de la información obtenida con el Message Inspector, disponible en la Fig. 5.25, se comprueba que el mensaje desencriptado no corresponde a los datos generados por el DUT (i.e. 00000000000000FE3E090D0503AB0000). Sabiendo que el MIC fue calculado correctamente debido a que no se reportó un error de MIC, sabemos que el problema está en la AppSKey usada por el DUT. Las últimas claves generadas en el proceso de Join están disponibles en la interfaz web.

5.5. Integración con F-Interop

La integración con el entorno de pruebas F-Iterop [4] fue realizada satisfactoriamente. Para ello fue necesario enviar la imagen docker del TAS para que los componentes que se encargan de instanciar sesiones en F-Interop puedan desplegar el servicio al momento de que un usuario elige la herramienta F-LoRa para realizar pruebas de conformidad. Adicionalmente, se especifica en un fichero de configuración las opciones que deben ser desplegadas para que el usuario pueda indicar los tests que desea ejecutar, así como información sobre la versión de la herramienta. Este fichero de configuración se encuentra en el directorio *finterop* del repositorio github con el código desarrollado.

Actualmente no se encuentra disponible para ejecutar la herramienta en la plataforma F-Interop, pero está disponible un video [23] con un demo de la integración en Octubre de 2018, año en que finalizó el proyecto F-Interop.

Luego de integrado el TAS con el entorno F-Interop se continuaron realizando mejoras y agregando herramientas permitiendo poder desplegar y ejecutar todo el entorno para pruebas de conformidad LoRaWAN en forma local con docker. Algunas de las herramientas desarrolladas, como el Message Inspector, también resultan de utilidad en escenarios donde el TAS se ejecuta de forma remota.

5.6. Resumen y Conclusión del Capítulo

En este capítulo se describieron las herramientas con las que cuenta el usuario para analizar la información relativa al intercambio de mensajes entre el DUT y el TAS en una sesión de test. Se especificaron los componentes de hardware y el

[4] `https://go.f-interop.eu/`

firmware con una implementación probada de LoRaWAN utilizado para probar la plataforma desarrollada. Luego de ejecutar los tests para este dispositivo y se mostró la información más relevante que ofrece la interfaz web, describiendo cómo se puede complementar esta información y en base a los timestamp de los mensajes visualizar el detalle de los mensajes en los logs el Message Inspector.

Para poder describir la forma en que un error en la implementación del DUT puede verse reflejado en los resultados de los tests se insertó errores mediante pequeñas modificaciones al firmware del DUT. Estos cambios fueron aplicados de forma independiente uno a uno y se mostró como cada uno de ellos era detectado por el TAS.

Finalmente se describió brevemente la integración de la plataforma desarrollada al entorno de tests F-Interop, proporcionando una referencia a un video con la demostración realizada en Octubre de 2018. Actualmente la plataforma F-Interop no está disponible para utilizarse, pero las herramientas adicionales desarrolladas posteriormente a la finalización de ese proyecto hacen posible y más conveniente ejecutar toda la plataforma en forma local.

El siguiente capítulo presenta las conclusiones generales del trabajo, resumiendo los principales resultados, aprendizajes y analizando algunas posibles líneas de trabajo futuro.

```
td_lorawan_act_01 PASS

td_lorawan_mac_01 PASS

td_lorawan_mac_02 Test td_lorawan_mac_02 failed with MACError error.

: No MAC command expected, but received:

: DevStatusAns MAC Command.

: Command ID (CID): 0x06

: Size: 3

: Content: 0xFE1D

: Battery Level: 0xFE

: Margin: 0x1D

:

: Test: td_lorawan_mac_02

: Step: S2NoMACCommandCheck

:

td_lorawan_reset PASS

td_lorawan_mac_03 PASS

td_lorawan_mac_04 PASS

td_lorawan_mac_05 PASS

td_lorawan_deactivate PASS
```

Figura 5.19: Reporte de error por no descartar un mensaje con comandos MAC simultáneamente en el FHDR y en el FRMPayload.

Figura 5.20: Mensaje enviado por el TAS con un comando MAC en el FHDR y otro en el FRMPayload. Este mensaje debe ser enviado según la especificación de LoRaWAN.

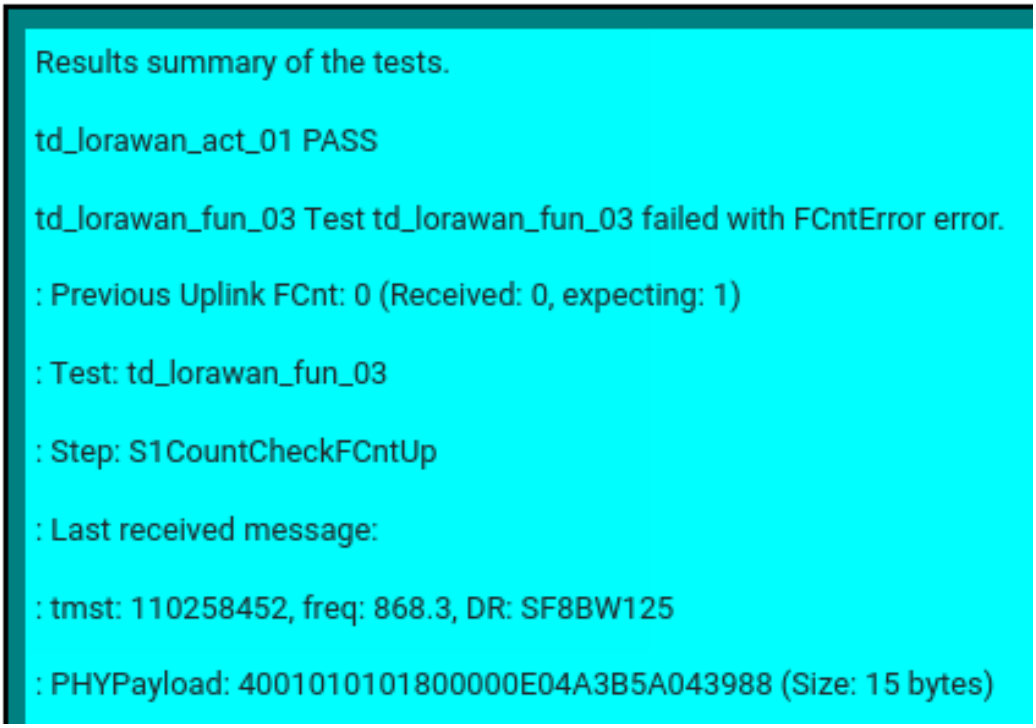

Figura 5.21: Reporte de error en el FCnt detectado por el test *td_lorawan_fun_03*.

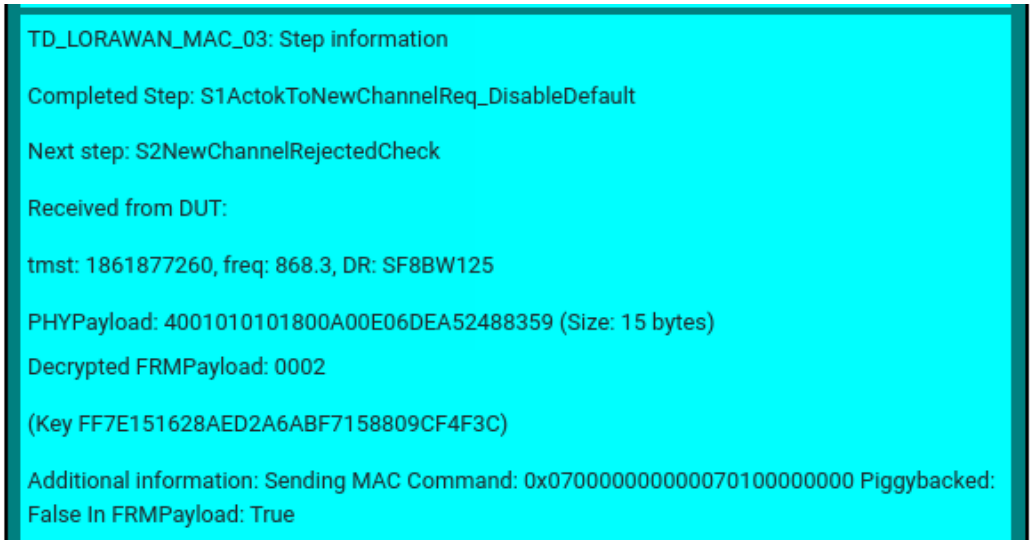

Figura 5.22: Información mostrada por la interfaz web al momento de recibir un mensaje del DUT y agendar un mensaje de Downlink con comandos para deshabilitar los primeros dos canales por defecto.

Figura 5.23: Mensaje recibido desde el DUT aceptando el cambio que solicitaba eliminar dos de los canales por defecto y por lo tanto debió ser rechazado.

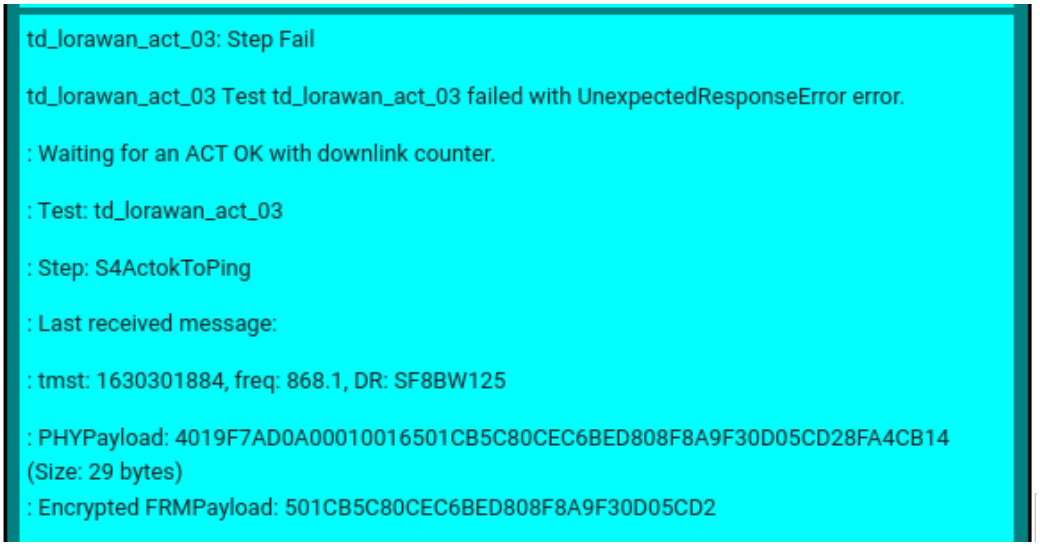

Figura 5.24: Error detectado al no recibir un TAOK confirmando que el DUT procesó el mensaje de downlink correspondiente y activó el modo de test..

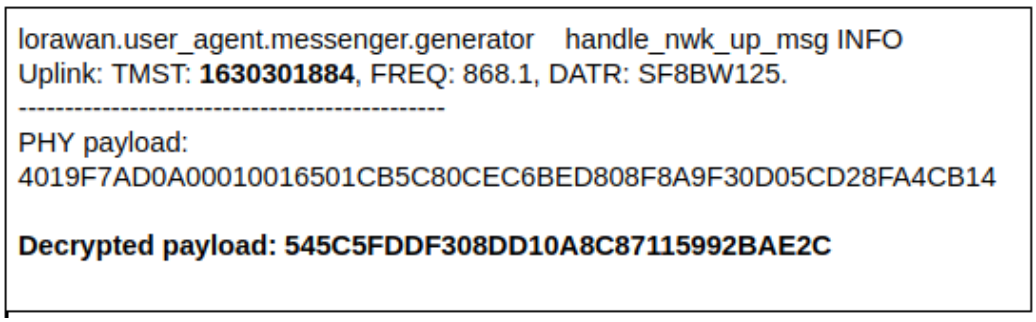

Figura 5.25: Información disponible en el Message Inspector donde se confirma que el contenido del mensaje no corresponde con el formato de datos enviados por el DUT.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Conclusiones

Este capítulo presenta las conclusiones generales, analizando cómo el diseño propuesto logró implementarse para cumplir con los objetivos planteados. Se muestran las principales dificultades encontradas, señalando aspectos que podrían haberse abordado de forma diferente y se sugieren posibles líneas de trabajo futuro.

Una vez probado el funcionamiento de la plataforma se puede afirmar que el resultado fue satisfactorio. Se logró implementar tests para las principales características de LoRaWAN en una plataforma que puede ser fácilmente ampliada para probar otras funcionalidades definidas en el protocolo de comunicación. El software desarrollado en este trabajo de tesis fue parte de un proceso de aprendizaje, y eso se ve reflejado en el hecho de que hay algunos aspectos del diseño que podrían mejorarse.

Se estudió en profundidad la especificación de LoRaWAN y se implementó la decodificación de los mensajes, así como los mecanismos de encriptación y de verificación de integridad de los mensajes. En base a esto se puede concluir que LoRaWAN logra combinar una especificación simple y fácil de implementar con mecanismos que permiten optimizar los parámetros de comunicación usados en la red y aprovechar las características de una capa física LoRa. Las nuevas características introducidas en LoRaWAN v1.1, versión de la especificación no incluida en este trabajo de tesis pero sobre la que se analizan las principales diferencias, parecen ir en el camino correcto de permitir escalabilidad facilitando a operadores de red implementar la interconexión de múltiples redes sobre la arquitectura definida.

Al evaluar el diseño propuesto y las posibilidades de crecimiento futuro que ofrece, se observa que la organización de los tests en pasos simples con pruebas básicas facilitó la reutilización de código, ayudó a identificar los motivos de un fallo en un test y además estos pasos implementados sirven de base para la implementación de nuevos tests. La elección de incorporar un Agente en el diseño, encapsulando en un componente simple y acotado la interacción con el Gateway LoRa, no sólo simplificó la integración de la plataforma con el entorno F-Interop sino que eventualmente permite desarrollar nuevas versiones compatibles con otros Gateways sin necesidad de realizar ninguna modificación en los demás componentes.

Una de los principales aciertos en la elección de la arquitectura y las tecnologías para la implementación de la plataforma fue basar la comunicación de los servicios en un broker de mensajes AMQP utilizando RabbitMQ. Tal como era esperable, tratándose de una arquitectura ampliamente utilizada, RabbitMQ fue clave para desacoplar el envío y recepción de los mensajes, permitiendo además la incorporación de herramientas adicionales para el análisis de los mensajes sin afectar los servicios centrales. Por otra parte, la utilización de Docker realizando el despliegue con la herramienta docker-compose demostró su enorme valor al momento de distribuir el software desarrollado. Un usuario puede ejecutar la plataforma simplemente instalando Docker, ejecutando simples comandos y todos los requerimientos y librerías necesarias vendrán incluidas y listas para usarse al descargarse automáticamente las imágenes de los servicios asociados a la plataforma, pudiendo configurar aspectos básicos del despliegue usando variables de entorno.

Al analizar la interfaz de usuario desarrollada la evaluación es positiva debido a que, a pesar de no ser un objetivo central del trabajo, se logró proporcionar una aplicación web que permite a un usuario interactuar con la plataforma de forma amigable. Sin embargo, es importante remarcar que el hecho de no estar desde un inicio dentro de los objetivos prioritarios se ve reflejado en la poca escalabilidad y flexibilidad para ampliar este componente. Se podría haber realizado un diseño con las mismas características básicas separando más claramente las responsabilidades de interacción con los componentes centrales y las de interacción con el usuario. De esta forma los módulos que se comunican con el TAS para intercambiar comandos y recibir información que debe ser mostrada al usuario podrían ofrecer una API REST a ser utilizada por el Web Front End. Un completo análisis y rediseño de la interfaz de usuario queda fuera del alcance de este trabajo de tesis y no es el objetivo perseguido en estas conclusiones.

Un claro punto a mejorar en la implementación de la plataforma está relacionado a la decodificación de los mensajes LoRaWAN, y en general a cómo se realiza la decodificación de todos los mensajes. La implementación actual cumple con las necesidades de una primera etapa de desarrollo y tiene las funcionalidades requeridas, pero por ejemplo en muchos casos se decodifican partes de mensajes antes de ser realmente utilizados. Si bien es un aspecto que debería ser analizado más en profundidad, una alternativa de diseño que parece interesante es separar la decodificación de los mensajes en un servicio independiente con una API REST bien definida. Una nueva iteración sobre este desarrollo debería optimizar la forma en que se manejan los mensajes evitando procesamiento innecesario antes de tiempo, y además es en estos módulos donde se percibe más claramente la necesidad de realizar tests unitarios al código que prueben las clases que decodifican mensajes.

En relación a la implementación de tests automatizados del código, además de implementar los mencionados tests unitarios, sería deseable contar con tests de integración para el código que prueben que cada prueba de conformidad implementada se integra correctamente a la plataforma. Esto facilitaría la implementación de nuevos tests, aprovechando las características modulares del diseño de la plataforma.

Para la visualización de los resultados y la identificación de posibles errores

en la implementación de LoRaWAN que se está sometiendo a pruebas resultaron fundamentales las herramientas auxiliares presentadas en la Sección 5.1. Una fortaleza del diseño implementado está en la utilización del sistema de manejo de excepciones de Python para reportar los errores detectados. Con esta estrategia cada test o componente que detecta un error simplemente tiene que lanzar la excepción correspondiente y son los componentes centrales de la plataforma quienes se encargan de procesar esta información y agregarla en el reporte final indicando la falla. Como contrapartida a esta flexibilidad y facilidad de reportar errores, el informe final presentado al usuario solo tiene información sobre el último mensaje recibido, y es por eso que tener la posibilidad de estudiar la secuencia de mensajes anteriores en base a los timestamp en el Message Inspector resulta fundamental en muchos casos.

En los resultados mostrados en el Capítulo 5 se puede apreciar el alcance global y el valor aportado por este trabajo de tesis. Sin quitar énfasis en la importancia que tendría la incorporación de tests de integración automatizados del código, las pruebas realizadas sobre un dispositivo ya certificado sirvieron no solo para ilustrar el funcionamiento de la plataforma sino también como tests manuales de que las funcionalidades de LoRaWAN fueron correctamente implementadas en el TAS. De cierta forma, estas pruebas permitieron comprobar que el inicio de sesión, los mecanismos de encriptación y las configuraciones de los parámetros de downlink, así como de las frecuencias usadas por el DUT fueron implementadas correctamente en el TAS.

Si bien no se prueban todos los errores posibles, ni se abordan todas las causas que pueden hacer fallar a cada uno de los tests implementados, la Sección 5.5 muestra cómo se comporta la plataforma ante la inyección de errores mediante modificaciones básicas en el DUT. Al analizar el comportamiento de la plataforma en este escenario vemos que cada uno de los errores inyectados fue detectado por todos los test que debían fallar y reportar el error, siendo este un fuerte argumento para afirmar que el TAS ha sido correctamente implementado. Sin lugar a dudas, más pruebas deberían ser realizadas si se quisiera ofrecer un producto terminado y que cumpla estándares mínimos de calidad, algo que claramente excede el alcance de este trabajo de tesis de maestría.

En cuanto a la integración con el entorno de tests del proyecto F-Interop, se destaca el valor de poder trabajar en coordinación con el grupo de desarrollo del mencionado proyecto, logrando que la plataforma de pruebas de conformidad LoRaWAN fuera integrada satisfactoriamente cómo la herramienta F-Lora, actualmente no disponible para ejecutarse en F-Interop. Analizando en perspectiva esta integración, y luego de haber realizado mejoras a la plataforma desarrollada y de agregar el soporte para la ejecución local, se puede afirmar que el caso de uso en el que un desarrollador pueda querer solo ejecutar el Agente de forma local y el TAS de forma remota no parece ser el escenario más probable. Un caso donde esta configuración remota podría ser interesante sería para aprovechar que el entorno remoto guarde un registro de los resultados y las sesiones ejecutadas. Alternativamente, esto puede ser fácilmente implementado en el entorno local como un servicio adicional y se menciona en las secciones siguientes dentro de las líneas de

posibles trabajos futuros.

6.1. Evaluación de los Objetivos Propuestos para la Tesis

Al observar los objetivos planteados al momento de iniciar este trabajo de tesis vemos que los mismos fueron alcanzados de forma satisfactoria. Sin dudas se logró comprender en profundidad el protocolo LoRaWAN, y esto se intentó reflejar en el Capítulo 2 que puede resultar de utilidad para alguien que quiera estudiar el funcionamiento de LoRaWAN, como complemento a la especificación de la LoRa Alliance.

En el Capítulo 3 y el Capítulo 4 se resume el proceso de análisis para el diseño e implementación, donde se han revisado varios conceptos relacionados a la arquitectura y despliegue de un proyecto de software. Esto además de aportar valor hacia las eventuales ampliaciones o trabajo futuro que se realice sobre la plataforma desarrollada, facilitando el mantenimiento y haciéndola escalable, logró satisfacer el objetivo de aprendizaje en esta área del desarrollo de software.

Luego del inicio de este trabajo de tesis, y mientras se desarrollaba el mismo, algunas empresas consolidadas han lanzado herramientas para realizar pruebas de pre-certificación. Ejemplos de estos son la plataforma ThingPark Connected [1] de Actility y la herramienta LoRaWAN Certification Test Tool (LCTT) [2] ofrecida a los miembros de la LoRa Alliance desde finales de 2019. El desarrollo de estas herramientas por un lado hacen que el trabajo de tesis realizado pueda perder interés para desarrolladores profesionales intentando llevar un producto al mercado, que sin dudas compensarán el pago de licencias con las ventajas que les ofrecen estas alternativas que incluyen pruebas completas de LoRaWAN en varias regiones. Por otra parte, el lanzamiento de estos productos pone de manifiesto la importancia y la necesidad de este tipo de herramientas y el hecho de que el desarrollo planteado inicialmente abordaba una necesidad real de muchos desarrolladores.

Un área donde se espera aportar valor y que la plataforma de conformidad LoRaWAN pueda ser utilizada es en los entornos educativos. La ampliación de la plataforma con nuevos tests puede disparar proyectos interesantes para comprender el funcionamiento de LoRaWAN.

En la siguiente sección se intentan aportar ideas para posibles trabajos futuros basados en el desarrollo realizado en la presente tesis de maestría.

6.2. Posibles Líneas de Trabajo Futuro

Se proponen algunas ideas para líneas de trabajo futuro desde la perspectiva de cuáles podrían ser los pasos a seguir si el el alcance de este trabajo fuera mayor y si se contara con más recursos para realizar una nueva iteración en el desarrollo. Se agrupan estas propuestas en:

[1] `https://partners.thingpark.com/en/thingpark-connected`
[2] `https://lora-alliance.org/lorawan-certification-test-tool`

- mejoras en la calidad del software ya desarrollado,
- nuevas funcionalidades para mejorar la experiencia del usuario y
- desarrollo de nuevos tests.

En lo que refiere a las mejoras a la calidad del software está la ya mencionada incorporación de tests de integración automatizados que permitan además probar que cada nuevo test realiza la transición entre los diferentes pasos tal como es esperado. También se deberían implementar tests unitarios para los pasos que son reutilizados, así como para los módulos que realizan la decodificación de los mensajes. Sin duda alguna, tratándose de una plataforma para realizar pruebas de conformidad, contar con niveles mínimos de calidad asegurados por tests automatizados en un proceso de integración continua sería algo muy positivo.

Se mencionó anteriormente la posibilidad de revisar los módulos de decodificación de mensajes LoRaWAN para optimizar la forma en la que estos son procesados y ahorrar tiempos de procesamiento cuando todavía no es necesario analizar partes del mensaje. Esto además de apuntar hacia un código más eficiente y fácil de mantener, podría jugar un rol importante en casos donde se esté ejecutando la plataforma de forma remota o análisis adicionales requieran más tiempo de procesamiento. Hay que tener en cuenta que el TAS debe analizar el mensaje recibido desde el DUT, procesar la respuesta y agendar el mensaje de downlink en el Gateway con tiempo suficiente para que este no sea rechazado. Hasta ahora esto no ha sido un problema y el tiempo medido desde que llega un mensaje de Uplink hasta que se agenda el mensaje de downlink correspondiente se ha medido y está en el entorno de los 10 a 30 ms (esta información es desplegada por el Agente en cada mensaje que se envía al DUT). En el ejemplo de la Sección 5.1 donde se mostraba que el fallo estaba relacionado a que el Gateway rechazó el mensaje por recibirlo en un tiempo incorrecto muestra que, si bien estos son casos aislados, esta situación puede ocurrir [3]

Algunas ideas para funcionalidades nuevas que mejoren la experiencia del usuario están relacionadas con el manejo de las sesiones de test y el almacenamiento de los resultados del mismo. Por ejemplo, sería deseable que la propia interfaz web permita dar inicio a la sesión de tests y lanzar la ejecución del mismo sin necesidad de coordinar esto con la ejecución de comandos para el despliegue de los componentes de la plataforma. Adicionalmente, mantener una base de datos con persistencia de las sesiones y almacenamiento de los reportes, tanto del TAS como de las herramientas auxiliares, serían funcionalidades que podrían mejorar la experiencia de un usuario.

Finalmente, otra línea de trabajo futuro ya mencionada y contemplada desde el inicio es la incorporación de nuevos tests. Estos pueden ser para probar más características definidas para la región Europea, como por ejemplo probar el funcionamiento del mecanismo Adaptive Data Rate explicado en la Subsección 2.5.2, o incluir tests para otras regiones como por ejemplo la AU915 utilizada en Uruguay.

[3] En el mencionado ejemplo de la Sección 5.1 el error reportado es "TOO_EARLY", pero es un error conocido del Packet Forwarder reportar "TOO_EARLY" cuando en realidad el mensaje se ha recibido demasiado tarde para agendar el downlink a tiempo.

6.3. Consideraciones Finales

El trabajo de tesis presentado significó un gran aprendizaje personal, donde no solo se tuvo que abordar el funcionamiento del protocolo LoRaWAN, la implementación de algunas de las funcionalidades de un servidor de red LoRaWAN y el estudio del diseño de pruebas de conformidad para un estándar, sino que los principales retos estaban relacionados con lograr una plataforma de software fácil de mantener, extensible y que aporte valor al ser reutilizada.

Una vez finalizado satisfactoriamente el trabajo, si se evalúa objetivamente el resultado obtenido y se lo compara con las alternativas desarrolladas por empresas consolidadas mencionadas en la Sección 6.1, se debe decir que la plataforma para pruebas de conformidad presentada posiblemente no resulte una opción atractiva para ser utilizada por equipos de desarrollo profesionales. Sin embargo, se considera que el trabajo realizado puede ser un modesto pero valioso aporte en entornos educativos. Por un lado puede aportar valor para realizar tests de una implementación de LoRaWAN teniendo control total de la plataforma de pruebas, y por otra parte de la potencial ampliación de la plataforma pueden surgir proyectos interesantes.

Apéndice A

Descripción de los tests de Conformidad

A.1. Activación (ACT)

A.1.1. Inicio de sesión con ABP

Tabla A.1: Descripción del test con ID td_lorawan_act_01.

Test ID	td_lorawan_act_01
Objetivo	Probar que el dispositivo puede enviar datos a la red usando el mecanismo de activación ABP.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene configurados los parámetros de DevAddr, NwkSKey y AppSkey, y estos son los mismos que están configurados en las sesión de test del TAS.
Paso	Descripción
1	▪ Recepción desde el DUT: Paquete de datos. ▪ TAS envía hacia el DUT: Mensaje de activación del Modo Test (payload 0x01010101 en el puerto 224, encriptado con la AppSKey).

Paso	Descripción
2	▪ Recepción desde el DUT: TAOK con el contador de downlink ▪ TAS envía hacia el DUT: -

A.1.2. Inicio de sesión con OTAA

Tabla A.2: Descripción del test con ID td_lorawan_act_02.

Test ID	td_lorawan_act_02
Objetivo	Probar que la activación de tipo OTAA está correctamente implementada. Cambiando los parámetros de DR de las ventanas de recepción de downlink (RX windows), se verifica que el DUT puede unirse a una red LoRaWAN usando OTAA en RX1 y en RX2.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT está configurado en Modo Test y soporta el mecanismo de OTAA.
Paso	**Descripción**
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: Trigger Join Request (test ID 0x06) indicando que el DUT debe solicitar una nueva session.
2	▪ Recepción desde el DUT: mensaje de Join Request. ▪ TAS envía hacia el DUT: mensaje de Join Accept configurando DLSettings (RX1DRoffset=2 y RX2DR=3).

Paso	Descripción
3	Un mensaje de aplicación con datos es esperado para luego activar el Modo Test del DUT. ▪ Recepción desde el DUT: mensaje de datos. ▪ TAS envía hacia el DUT: Mensaje de activación del Modo Test en el DUT (mensaje de DL con payload 0x01010101 hacia el puerto 224)
4	Se espera a la recepción del siguiente mensaje de ACTOK con el contador de downlink y como respuesta se envía un mensaje de PING en RX1 usando el DR offset configurado durante la activación. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: PING en RX1 usando DR offset igual a 2.
5	Espera por el PONG correspondiente al mensaje anterior y envía un nuevo mensaje de PING pero ahora usando RX2. ▪ Recepción desde el DUT: PONG. ▪ TAS envía hacia el DUT: PING en RX2.
6	Verificación del PONG recibido como respuesta. ▪ Recepción desde el DUT: PONG message. ▪ TAS envía hacia el DUT: -

A.1.3. Modificación de ventanas de downlink durante la activación

Tabla A.3: Descripción del test con ID td_lorawan_act_03.

Test ID	td_lorawan_act_03
Objetivo	Pobar la modificación de los tiempos en los que se inician las ventanas de Downlink durante el proceso de activación OTAA. Se verifica que el nodo puede activarse en ambas ventanas (RX1 y RX2).
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: Trigger Join Request (test ID 0x06) indicando que el DUT debe solicitar una nueva session.
2	▪ Recepción desde el DUT: Mensaje de Join Request. ▪ TAS envía hacia el DUT: mensaje de Join Accept configurando RxDelay=3 seg.
3	Se espera la recepción de un mensaje de datos para activar nuevamente el Modo Test. ▪ Recepción desde el DUT: mensaje de datos. ▪ TAS envía hacia el DUT: Mensaje de activación del Modo Test en el DUT (mensaje de DL con payload 0x01010101 hacia el puerto 224)

Paso	Descripción
4	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de tipo PING (id 0x4) en la ventana RX1 (usando los 3 segundos configurados previamente).
5	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: nuevo mensaje de PING, ahora usando la ventana RX2.
6	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: -

A.1.4. Adición de canales durante la activación

Tabla A.4: Descripción del test con ID td_lorawan_act_04.

Test ID	td_lorawan_act_04
Objetivo	Configurar 5 nuevos canales durante el proceso de activación OTAA.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: Trigger Join Request (test ID 0x06) indicando que el DUT debe solicitar una nueva session.
2	▪ Recepción desde el DUT: mensaje de Join Request. ▪ TAS envía hacia el DUT: Join Accept configurando nuevos canales con el campo CFList.
3	▪ Recepción desde el DUT: mensaje de datos. ▪ TAS envía hacia el DUT: Mensaje de activación del Modo Test en el DUT (mensaje de DL con payload 0x01010101 hacia el puerto 224).

Paso	Descripción
4	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

A.1.5. Renovación de sesión

Tabla A.5: Descripción del test con ID td_lorawan_act_05.

Test ID	td_lorawan_act_05
Objetivo	Iniciar una nueva sesión en el DUT reestableciendo los parámetros MAC por defecto de LoRaWAN.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: Trigger Join Request (test ID 0x06) indicando que el DUT debe solicitar una nueva session.
2	▪ Recepción desde el DUT: mensaje de Join Request. ▪ TAS envía hacia el DUT: mensaje de Join Accept.

Paso	Descripción
3	▪ Recepción desde el DUT: mensaje de datos. ▪ TAS envía hacia el DUT: Mensaje de activación del Modo Test en el DUT (mensaje de DL con payload 0x01010101 hacia el puerto 224).
4	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de ping usando RX1 con los parámetos por defecto.
5	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: mensaje de PING usando RX2 con los parámetros por defecto.
6	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: -

A.2. Funcionalidades básicas y tiempos (FUN)

A.2.1. Intercambio básico de mensajes

Tabla A.6: Descripción del test con ID td_lorawan_fun_01.

Test ID	td_lorawan_fun_01
Objetivo	Verifica el intercambio básico de mensajes con una secuencia de PING-PONG.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de PING.
2	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: -
3	Verifica que se reciben correctamente dos mensajes TAOK. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

A.2.2. Tolerancia a errores de tiempo en la recepción

Tabla A.7: Descripción del test con ID td_lorawan_fun_02.

Test ID	td_lorawan_fun_02
Objetivo	Comprueba la tolerancia del DUT a errores de tiempo en la recepción de mensajes de Downlink. Se espera que los mensajes sean recibidos con errores de +/20 microsegundos en el instante de envío, tanto para la ventana RX1 como para la ventana RX2.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de PING en la ventana de Downlink RX1 enviado 20 microsegundos tarde.
2	Recepción desde el DUT: respuesta PONG. TAS envía hacia el DUT: - ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de PING en la ventana de Downlink RX1 enviado 20 microsegundos tarde.
3	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de PING en la ventana de Downlink RX2 enviado 20 microsegundos tarde.

Paso	Descripción
4	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: -
5	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de PING en la ventana de Downlink RX1 enviado 20 microsegundos antes de lo esperado.
6	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: -
7	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de PING en la ventana de Downlink RX2 enviado 20 microsegundos antes de lo esperado.
8	▪ Recepción desde el DUT: respuesta PONG. ▪ TAS envía hacia el DUT: -

A.2.3. Manejo de números de secuencia

Tabla A.8: Descripción del test con ID td_lorawan_fun_03.

Test ID	td_lorawan_fun_03
Objetivo	Verifica los números de secuencia en los mensajes de Uplink.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	Se espera recibir una secuencia de tres mensajes de TAOK verificando que el FCnt aumente correctamente. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

A.2.4. Números de secuencia decrecientes

Tabla A.9: Descripción del test con ID td_lorawan_fun_04.

Test ID	td_lorawan_fun_04
Objetivo	Verifica que un mensaje con el FCnt decreciente es ignorado.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de desactivación del Modo Tests pero usando un FCnt menor al último enviado, el mensaje debe ser ignorado.
2	Verifica que el mensaje enviado anteriormente sea ignorado y por lo tanto no se incremente el contador de downlink. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

A.2.5. Mensajes con confirmación

Tabla A.10: Descripción del test con ID td_lorawan_fun_05.

Test ID	td_lorawan_fun_05
Objetivo	Prueba la impolementación de los mensajes con confirmación, verificando que el bit de ACK de los mensajes de uplink se maneja correctamente.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: activación de mensajes con confirmación para uplink (ID 0x02 en el Payload).
2	▪ Recepción desde el DUT: TAOK en un mensaje con confirmación. ▪ TAS envía hacia el DUT: activación de mensajes sin confirmación para el uplink (ID 0x03 en el Payload).

A.2.6. Retransmisión de mensajes no reconocidos

Tabla A.11: Descripción del test con ID td_lorawan_fun_06.

Test ID	td_lorawan_fun_06
Objetivo	Verifica que un mensaje de uplink con confirmación es reenviado si no es reconocido (ACK) por el servidor de red.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: activación de mensajes con confirmación para uplink (ID 0x02 en el Payload).
2	El TAS no envía reconocimiento de los mensajes y espera recibibr 2 retransmisiones del TAOK con confirmación. ▪ Recepción desde el DUT: TAOK en un mensaje con confirmación. ▪ TAS envía hacia el DUT: activación de mensajes sin confirmación para el uplink (ID 0x03 en el Payload).

A.3. Seguridad (SEC)

A.3.1. Verificación de la encriptación del payload

Tabla A.12: Descripción del test con ID td_lorawan_sec_01.

Test ID	td_lorawan_sec_01
Objetivo	Comprueba la implementación del mecanismos de encriptación AES mediante el intercambio de mensajes de PING-PONG. Adicionalmente se verifica que el MIC es calculado correctamente.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje de PING.
2	Inicia un intercambio de 10 mensajes PING-PONG. ▪ Recepción desde el DUT: respuesta PONG al último PING enviado. ▪ TAS envía hacia el DUT: nuevo mensaje PING de largo aleatorio.
3	▪ Recepción desde el DUT: verifica la última respuesta PONG. ▪ TAS envía hacia el DUT: -

A.3.2. Filtrado de mensajes con código de integridad incorrecto

Tabla A.13: Descripción del test con ID td_lorawan_sec_02.

Test ID	td_lorawan_sec_02
Objetivo	Comprueba que un mensaje sea ignorado si su MIC no es correcto.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje PING con el MIC erroneo.
2	Verifica que el PING anterior sea ignorado. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

A.4. Comandos MAC (MAC)

A.4.1. Intercambio básico de comandos

Tabla A.14: Descripción del test con ID td_lorawan_mac_01.

Test ID	td_lorawan_mac_01
Objetivo	Verifica el envío de un comando MAC DevStatus, solicitando el estado del DUT, en el campo FOpts.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: comando DevStatusReq en el campo FOpts.
2	▪ Recepción desde el DUT: mensaje de TAOK con la respuesta el comando MAC (DevStatusAns). ▪ TAS envía hacia el DUT: -
3	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: comando DevStatusReq en el FRMPayload (usando FPort 0).

Paso	Descripción
4	▪ Recepción desde el DUT: mensaje de TAOK con la respuesta el comando MAC (DevStatusAns). ▪ TAS envía hacia el DUT: -

A.4.2. Filtrado de mensajes con comandos simultáneamente en FOpts y Payload

Tabla A.15: Descripción del test con ID td_lorawan_mac_02.

Test ID	td_lorawan_mac_02
Objetivo	Verifica que un mensaje conteniendo comandos MAC en el FRMPayload y en el campo FOpts es ignorado.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje con dos comandos DevStatus-Req, uno en el FRMPayload y otro en FOpts.
2	Se verifica que el TAOK no contenga respuesta a los comandos anteriores, ya que el mensaje debió ser ignorado. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

Paso	Descripción
3	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink ▪ TAS envía hacia el DUT: mensaje con dos comandos DevStatus-Req, uno en el FRMPayload y otro en FOpts.
4	Se verifica que el TAOK no contenga respuesta a los comandos anteriores, ya que el mensaje debió ser ignorado. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

A.4.3. Verificación de que no se eliminan los canales por defecto

Tabla A.16: Descripción del test con ID td_lorawan_mac_03.

Test ID	td_lorawan_mac_03
Objetivo	Verifica que el DUT no deja de usar un canal por defecto a pesar de recibir la solicitud en un comando MAC.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: comando MAC NewChannelReq desabilitando uno de los canales por defecto.

Paso	Descripción
2	Verifica que se recibe una respuesta rechazando el cambio en las frecuencias. ▪ Recepción desde el DUT: mensaje con la respuesta NewChannelAns y estatus NO OK. ▪ TAS envía hacia el DUT: -
3	Se comprueba que todas las frecuencias siguen siendo utilizadas. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

A.4.4. Adición de un nuevo canal de comunicación

Tabla A.17: Descripción del test con ID td_lorawan_mac_04.

Test ID	td_lorawan_mac_04
Objetivo	Utiliza varios comandos MAC en un mismo mensaje para configurar canales adicionales, verificando luego que estos son utilizados.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	**Descripción**
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje con comandos MAC NewChannelReq configurando tres nuevos canales.

Paso	Descripción
2	Verifica que los comandos enviados sean aceptados. ▪ Recepción desde el DUT: mensaje con las respuestas NewChannelAns aceptando las nuevas frecuencias. ▪ TAS envía hacia el DUT: -
3	Elimina los canales añadidos anteriormente. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje con comandos NewChannelReq eliminando los canales añadidos en el paso anterior.
4	Verifica que los comandos enviados sean aceptados. ▪ Recepción desde el DUT: mensaje con las respuestas NewChannelAns aceptando las modificaciones. ▪ TAS envía hacia el DUT: -

A.4.5. Adición de más de un canal en el mismo mensaje

Tabla A.18: Descripción del test con ID td_lorawan_mac_05.

Test ID	td_lorawan_mac_05
Objetivo	Usa un comando MAC NewChannelReq para agregar una nueva frequencia, verifica que el cambio sea aceptado y que luego la nueva frecuencia sea usada.
Referencias	LoRaWAN Specification v1.0.2.
Pre-condiciones del test	El DUT tiene una sesión activa con el TAS y está configurado en Modo Test.
Paso	Descripción
1	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje con comando MAC NewChannelReq configurando un nuevo canal.
2	▪ Recepción desde el DUT: mensaje con respuesta NewChannelAns aceptando el cambio ▪ TAS envía hacia el DUT: -
3	Verifica que todas las frecuencias sean utilizadas, incluyendo la del canal añadido en el paso anterior. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

Paso	Descripción
4	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: mensaje con comando mac NewChannelReq eliminando el canal añadido previamente.
5	▪ Recepción desde el DUT: mensaje con respuesta NewChannelAns aceptando el cambio ▪ TAS envía hacia el DUT: -
6	Verifica que la frecuencia removida no sea utilizada. ▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -
7	▪ Recepción desde el DUT: mensaje de TAOK con el contador de downlink. ▪ TAS envía hacia el DUT: -

Apéndice B

Creación de servicios y Despliegue

B.1. Guía rápida para el inicio de una sesión de tests

Una vez clonado el repositorio de la plataforma [1] e instalado Docker y docker-compose [2], se pueden seguir estos pasos para ejecutar una sesión de tests:

1. Configurar el Gateway LoRa para que el Packet Forwarder envíe los mensajes al PC que ejecutará la plataforma (el servicio Agent será quien reciba sus mensajes).
    - Por defecto el Agent recibe los mensajes en el puerto UDP 1700, pero esto se puede modificar definiendo la variable de entorno *AGENT_PORT*.
    - Existen tutoriales para configurar el servicio de Packet Forwarder en los diferentes modelos de Gateway LoRa, algunos ejemplos pueden encontrarse en la web de un proveedor de red LoRaWAN ampliamente utilizado como The Things Network [3].
2. Iniciar la interfaz de usuario y el broker de mensajes (RabbitMQ).
    - `make bootstrap_session_interface`
    - Abrir un navegador para visualizar la interfaz web:
        - `https://localhost:8081`
3. Iniciar el TAS
    - `make launch_test_session`
    - Una vez el TAS se ha iniciado correctamente el botón *Config* se habilita en la interfaz web.

[1] `https://github.com/pablomodernell/lorawan_conformance_testing`
[2] `https://docs.docker.com/compose/install/`
[3] `https://enterprise.thethingsstack.io/gateways/`

4. Configurar la sesión eligiendo los test a ejecutar e indicar las claves ABP configuradas en el dispositivo LoRaWAN que se someterá a pruebas.
    - Seleccionar los tests a ejecutarse en la interfaz de usuario y presionar el botón *Configure*.
    - Indicar la dirección (devAddr) asignada y claves de sesión ABP configuradas en el dispositivo y presionar el botón *Personalize*.
5. (Opcional y recomendado) Iniciar el servicio auxiliar *Message Inspector*
    - `make start_inspector`
    - Para visualizar los logs de este servicio:

      ```
      docker logs $(docker-compose ps -q agent-mock) -f
      ```
6. Iniciar el Agent
    - `make start_agent`
    - Para visualizar los logs de este servicio:

      ```
      docker logs $(docker-compose ps -q agent) -f
      ```
7. Comenzar la sesión de test en la interfaz web
    - Presionar el botón *Start Session*
8. Encender el dispositivo LoRaWAN para que comience a enviar mensajes.

B.2. Despliegue de servicios

El despliegue de la plataforma se centraliza en los ficheros

- *docker-compose.yml*: donde se definen los servicios y se indica qué imagen debe usarse para crear cada uno de ellos. Salvo que se haya creado una nueva imagen y exista localmente, las imágenes se descargan del repositorio de Docker [4]
- *Makefile*: fichero que facilita la creación de imágenes docker a partir del código y la publicación de nuevas versiones de las imágenes.
    - Por ejemplo, una vez realizadas modificaciones al código se puede actualizar la versión en la varible VERSION de este fichero y crear una nueva versión con el comando:

      ```
      make build_coformance
      ```

[4] `https://hub.docker.com/`

- *.env*: este fichero contiene variables de entorno utilizadas por docker-compose al momento de desplegar los servicios. Estos parámetros se pueden sobreescribir definiendo una variable de entorno con el mismo nombre, por ejemplo para especificar una versión diferente de un componente o para indicar un puerto diferente para la interfaz web o para el Agent.

B.3. Agente de Usuario y CLI

Es posible enviar mensajes LoRaWAN simulados al TAS, sin necesidad de conectar un Gateway LoRa y un dispositivo LoRaWAN. Para esto es necesario, una vez iniciada una sesión de test, lanzar el servicio auxiliar *Message Inspector* y luego abrir una consola de comandos con el servicio CLI:

- `make start_inspector`
- Para visualizar los logs de este servicio:

  ```
  docker logs $(docker-compose ps -q agent-mock) -f
  ```

- `make open_cli`
- En esta consola están disponibles para ejecutarse todos los comandos que se describieron en la Sección 5.1.

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[1] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the limits of lorawan. *IEEE Communications magazine*, 55(9):34–40, 2017.

[2] LoRa Alliance. Lorawan 1.0. 3 regional parameters 2018.

[3] LoRa Alliance. F-interop project aims and objectives. `https://f-interop.eu/index.php/about-f-interop/aims-and-objectives`, visitado Junio 2020.

[4] LoRa Alliance. Lora alliance members directory. `https://lora-alliance.org/member-directory`, visitado Junio 2020.

[5] LoRa Alliance. Lorawan security whitepaper. `https://lora-alliance.org/resource-hub/lorawan-security-whitepaper`, visitado Junio 2020.

[6] Lora Alliance. What is lorawan. `https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf`, visitado Junio 2020.

[7] LoRa Alliance. Why lorawan is the for asset-tracking connectivity. `https://lora-alliance.org/sites/default/files/2020-05/WhitePaper_AssetTracking.pdf`, visitado Junio 2020.

[8] LoRa Alliance. Why lorawan is the foundation for smart building success. `https://lora-alliance.org/sites/default/files/2020-05/LA_WhitePaper_SmartBuildings_0520_v1.1_1.pdf`, visitado Junio 2020.

[9] LoRa Alliance. Why lorawan is the logical choice for crop production monitoring solutions. `https://lora-alliance.org/sites/default/files/2020-05/LA_WhitePaper_SmartAg_0520_v2.pdf`, visitado Junio 2020.

[10] MURATA ELECTRONICS EUROPE BV. Cmwx1zzabz-078 certification. open mcu lorawan module for eu. `https://lora-alliance.org/sites/default/files/2018-06/16093001_001_lorawan_1.0.1_-_test_report_-_murata_cmwx1zzabz-078.pdf`, visitado Julio de 2020.

[11] CEPT Electronic Communications Committee et al. Erc/rec 70-03. *Relating to the use of short range devices (SRD)*, 2013.

[12] Tektelic Communications. Kona micro iot gateway. https://tektelic.com/wp-content/uploads/KONA-Micro.pdf, visitado Julio de 2020.

[13] ETSI ETSI ES. 202 237: Methods for testing and specification (mts); internet protocol testing (ipt); generic approach to interoperability testing. *European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France*, 2007.

[14] IoT-Analytics. Lpwan market report 2018-2023. https://iot-analytics.com/product/lpwan-market-report-2018-2023/, visitado Junio 2020.

[15] International Telecommunication Union ITU-T. Recommendation itu-t y.2060: Overview of the internet of things. https://www.itu.int/rec/T-REC-Y.2060-201206-I, visitado Junio 2020.

[16] Haider Mahmood Jawad, Rosdiadee Nordin, Sadik Kamel Gharghan, Aqeel Mahmood Jawad, and Mahamod Ismail. Energy-efficient wireless sensor networks for precision agriculture: A review. *Sensors*, 17(8):1781, 2017.

[17] I. E. Korbi, Y. Ghamri-Doudane, and L. A. Saidane. Lorawan analysis under unsaturated traffic, orthogonal and non-orthogonal spreading factor conditions. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–9, 2018.

[18] Link Labs. A comprehensive look at low power, wide area networks for internet of things engineers and decision makers. https://www.link-labs.com/lpwan, visitado Junio 2020.

[19] Murata Manufacturing. Murata sub-g module data sheet. https://wireless.murata.com/datasheet?/RFM/data/type_abz.pdf, visitado Julio de 2020.

[20] Borja Martínez, Ferran Adelantado, Andrea Bartoli, and Xavier Vilajosana. Exploring the performance boundaries of nb-iot. *CoRR*, abs/1810.00847, 2018.

[21] ST Microelectronics. Stm32 lora expansion package for stm32cube. https://www.st.com/resource/en/user_manual/dm00300436-stm32-lora-expansion-package-for-stm32cube-stmicroelectronics.pdf. visitado Julio de 2020.

[22] ST Microelectronics. Discovery kit for lorawan, sigfox, and lpwan protocols with stm32l0. https://www.st.com/resource/en/data_brief/b-l072z-lrwan1.pdf, visitado Julio de 2020.

[23] Pablo Modernell. F-lora: Lorawan conformance testing for f-interop. https://www.youtube.com/watch?v=mVxw0p4qSGQ&t=1s, visitado Julio de 2020.

[24] International Standards Organization. Iso/iec 9646-1:1994 information technology — open systems interconnection — conformance testing methodology and framework — part 1: General concepts. https://www.iso.org/obp/ui/#iso:std:iso-iec:9646:-1:ed-2:v1:en, visitado Junio 2020.

[25] Maria Rita Palattella, Federico Sismondi, Tengfei Chang, Loïc Baron, Mališa Vučinić, Pablo Modernell, Xavier Vilajosana, and Thomas Watteyne. F-interop platform and tools: Validating iot implementations faster. In *International Conference on Ad-Hoc Networks and Wireless*, pages 332–343. Springer, 2018.

[26] Maria Rita Palattella, Xavier Vilajosana, Tengfei Chang, Miguel Angel Reina Ortega, and Thomas Watteyne. Lessons learned from the 6tisch plugtests. In *International Internet of Things Summit*, pages 415–426. Springer, 2015.

[27] Ramon Sanchez-Iborra and Maria-Dolores Cano. State of the art in lp-wan solutions for industrial iot services. *Sensors*, 16(5):708, May 2016.

[28] Semtech. Etsi compliance of the sx1272/3 lora modem. https://semtech.my.salesforce.com/sfc/p/E0000000JelG/a/44000000MDmi/goMicMOm664IJPTgp5JP4Vn5mf1zAQmAQEhUq5rsuLc?__hstc=212684107.ad863e0cd3d40c95e8154288608bda89.1589048848775.1592245224905.1592772433392.10&__hssc=212684107.1.1592772433392&__hsfp=3350528426, visitado Junio 2020.

[29] AN Semtech. 120022, lora modulation basics, may, 2015.

[30] N Sornin and A Yegin. Lorawan 1.0. 3 specifications. *LoRa Alliance, Ver*, 1(3):1–72, 2018.

[31] Statista. Number of lpwan connections by technology worldwide from 2017 to 2023. https://www.statista.com/statistics/880822/lpwan-ic-market-share-by-technology/, visitado Junio 2020.

[32] Ovidiu Vermesan, Peter Friess, et al. *Internet of things-from research and innovation to market deployment*, volume 29. River publishers Aalborg, 2014.

[33] Sebastien Ziegler, Serge Fdida, Thomas Watteyne, and César Viho. F-interop-online conformance, interoperability and performance tests for the iot. 2016.

Esta página ha sido intencionalmente dejada en blanco.

Glosario

ABP Activation By Personalization. 21, 40, 47

ADR Adaptive Data Rate. 26, 27, 28, 29

AMQP Advanced Messaging Queueing Protocol. 56, 59, 61, 69, 71, 72, 74

AppKey Application Key. 22, 23, 24, 34, 35

AppSKey Application Session Key. 17, 19, 20, 21, 35

BW Bandwidth/Ancho de banda. 10

CSS Chirp Spread Spectrum. 9, 12

DC Duty Cycle. 11, 14

DR Data Rate. 10, 13, 15, 17, 25, 27, 28, 31, 32

DUT Device Under Test. 40, 41, 42, 43, 45, 46, 47, 50, 51, 52, 60, 61, 63, 64, 65, 66, 69, 71, 72, 73, 74, 78, 79, 81, 82, 83, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98

FHDR Frame Header. 20, 25, 29, 33

FPort Frame Port. 20, 29

FRMPayload Frame Payload. 20, 21, 29, 31

IoT Internet of Things. 1, 4

LPWAN Low Power Wide Area Networks. 1, 2, 12

MHDR MAC Header. 19, 20, 23, 24, 26

MIC Message Integrity Code. 6, 19, 20, 23, 24, 33, 35, 36, 40, 64, 87, 88, 90, 97

NB-IoT Narrow Band Internet of Things. 1, 2

NwkKey Network Key. 34, 35, 36

NwkSKey Network Session Key. 17, 19, 20, 21, 29

OTAA Over The Air Activation. 21, 34, 40, 47, 50, 65, 66

PF Packet Forwarder. 69, 71, 72

SF Spreading Factor. 10, 13, 17

TAOK Test Activation OK. 46, 50, 51, 84, 88, 94, 97

TAS Test Application Server. 41, 42, 43, 44, 45, 46, 47, 52, 53, 56, 61, 62, 66, 69, 71, 72, 73, 74, 77, 78, 79, 80, 81, 83, 84, 87, 88, 95, 96, 97, 98, 148

TSC Test Session Coordinator. 60, 61, 65, 74

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el jueves 15 octubre, 2020.
http://iie.fing.edu.uy/