Integración de un modelo de energía en la simulación de redes LoRaWAN en NS-3

Gabriel Dobato Ingalaturre
Máster Universitario de Ingeniería de Telecomunicación
Telemática

Jose Lopez Vicario
Xavier Vilajosana Guillen

10 de Junio 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	Integración de un modelo de energía en la simulación de redes LoRaWAN en NS-3
Nombre del autor:	Gabriel Dobato Ingalaturre
Nombre del consultor/a:	Jose Lopez Vicario
Nombre del PRA:	Xavier Vilajosana Guillen
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	Máster Universitario en Ingeniería de Telecomunicación
Área del Trabajo Final:	Telemática
Idioma del trabajo:	Castellano
Palabras clave	LoRaWAN LPWAN NS-3

Resumen del Trabajo

En el contexto de *Internet of Things,* LoRaWAN se ha posicionado en los últimos años como unas de las tecnologías más prominentes dentro de las redes LPWAN dada sus altas prestaciones de alcance y consumo energético, despertando con ello el interés de la comunidad investigadora y desarrolladora en este ámbito. Sin embargo, debido su reciente irrupción y la propiedad de la tecnología LoRa, todavía es limitada la información que puede encontrarse en la literatura actual acerca de su rendimiento en escenarios reales con una densidad de dispositivos comparable al que contempla su especificación, surgiendo con ello la necesidad de crear entornos simulados para llevar a cabo tales medidas de rendimiento.

En el presente trabajo se lleva a cabo la integración de un modelo de energía en un módulo de simulación de LoRaWAN implementado bajo el entorno de simulación de redes NS-3, permitiendo con ello el análisis del consumo energético de los dispositivos que operan bajo alimentación autónoma en la red. Para tal fin, se ha diseñado un modelo teórico de consumo y de fuente de alimentación autónoma de los dispositivos finales de red, el cual ha sido codificado en C++ haciendo uso del *framework* de energía de NS-3, de acuerdo a un diseño y arquitectura de software definidas, y siguiendo un proceso de validación que verifica una serie de métricas establecidas.

El trabajo contempla la creación de escenarios simulados para la aplicación del modelo energético, emulando entornos con visión directa entre dispositivos

(LOS), así como entornos urbanos sin visión directa (NLOS) con una densidad variable de dispositivos y haciendo uso de diferentes topologías, para realizar un posterior análisis de acuerdo a los resultados obtenidos en los mismos.

Abstract:

Within context of the Internet of Things, LoRaWAN has positioned itself as one of the most remarkable technologies within the LPWAN networks due to its performance of range and energy consumption, arousing the interest of the research and development community in this area. However, due to its recent irruption and the ownership of the LoRa technology, the information that can be found in the current literature about its performance in real scenarios with a density of nodes comparable to which the specification considers, is still limited. Thus, emerging the need to create simulated environments to carry out such performance measures.

In the present dissertation, the integration of an energy model is carried out in a LoRaWAN simulation module, implemented in the networks simulation NS-3, thus allowing the energy consumption analysis of battery-operated devices in the network. For that purpose, a theoretical model of device consumption and of an energy source of the network end-devices has been designed and codified in C++, by using the NS-3 energy framework, according to a defined software design and architecture, and by following a validation process that verifies a series of established metrics.

The dissertation includes the creation of simulated scenarios to use the designed energy model, emulating Line-of-sight environments, as well as urban environments, where a variable density of devices and different topologies has been considered, in which different data has been collected for further analysis.

Agradecimientos

A Jenifer, por su apoyo incondicional
en este largo camino.

A Jose Lopez, por todo el soporte recibido.

Índice

Índice de tablas y figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

En los últimos años el concepto de *Internet of Things* (IoT) ha irrumpido con gran fuerza en diferentes ámbitos de la sociedad actual formando junto con el *Big Data* una dupla tecnológica ideal que ha dado paso a nuevas formas de entender la interconexión de redes y la gestión de la información, convirtiéndose en un proceso disruptivo que amplía el alcance de digitalización a ámbitos inimaginables hace unos cuantos años atrás.

Varias son las tecnologías que han emergido para dar respuesta a los nuevos desafíos tecnológicos que subyacen de este nuevo paradigma, operando con diferentes velocidades de datos, desde unos pocos bps hasta Mbps, con rangos que oscilan entre unos cuantos metros hasta más de un kilómetro, bajo bandas frecuenciales licenciadas o no licenciadas, etc., constituyendo un conjunto de redes heterogéneas que operan bajo un marco común. En este contexto, las redes LPWAN (*Low Power Wide Area Network*), surgen como respuesta a aplicaciones que requieren una alta eficiencia energética, así como un amplio alcance para dar cobertura a dispositivos que operan bajo diferentes escenarios IoT. Tecnologías como Sigfox, Ingenu y LoRa compiten actualmente para predominar en el ámbito de las redes LPWAN y ganar una posición de relevancia en el mercado IoT.

Bajo este marco, LoRa parece haberse posicionado como una de las tecnologías más prominentes dentro de las redes LPWAN debido a sus altas prestaciones de alcance, velocidad de datos, uso del espectro, seguridad y eficiencia energética, que satisfacen ampliamente los requerimientos técnicos que caracterizan a las redes que constituyen el paradigma IoT para dar servicio a los diferentes escenarios de conectividad de larga distancia.

Aunque LoRa es una tecnología propietaria, correspondiente a la capa física del *stack,* el resto del protocolo está libre de patentes y recibe el continuo impulso de la asociación abierta y sin ánimo de lucro LoRa Alliance, y la especificación LoRaWAN [24] que surge de ella, correspondiente a la capa de enlace del *stack,* con el objetivo de fomentar esta tecnología y garantizar la interoperabilidad entre operadores en un estándar global abierto. Como consecuencia, durante los últimos años, varios son los estudios que se han llevado a cabo y que están presentes en la literatura actual, con el fin de medir el rendimiento y la operabilidad de la red estableciendo diferentes métricas según sea el objeto de estudio. Entre las prestaciones teóricas que proporciona LoRaWAN, hay una que llama especialmente la atención a gran parte de la comunidad investigadora y al conjunto de desarrolladores bajo el contexto IoT. Se trata del alto rendimiento de consumo y eficiencia energética de los dispositivos que operan en la red, desafiando a otras tecnologías con autonomías que pueden llegar hasta los 10 años haciendo uso de sistemas de baterías estándar.

Debido a la reciente irrupción de la tecnología, no resulta sencillo llevar a cabo estudios relacionados con la eficiencia energética de los dispositivos bajo escenarios reales, caracterizados por la interconexión de miles de nodos. Por ello, los trabajos que pueden encontrarse en la literatura actual están sujetos a ciertas limitaciones logísticas, y particularizan el estudio a situaciones específicas y con un número limitado de dispositivos.

En este contexto, el presente Trabajo Final de Máster (TFM) intenta dar solución a la limitación anterior, permitiendo que el análisis de eficiencia energética en redes LoRaWAN pueda llevarse a cabo bajo escenarios de alta densidad de dispositivos. En este sentido, el presente trabajo tiene pretende integrar un modelo de energía bajo un entorno simulado de red LoRaWAN que permita llevar a cabo un análisis de la eficiencia energética de los dispositivos bajo escenarios de alta densidad. Para ello, se hará uso del entorno de simulación de redes basado en eventos discretos NS-3, y se tomará como punto de partida el módulo "LoRaWAN" desarrollado sobre la plataforma disponible en [4], en el cual se llevará a cabo la integración del modelo energético.

1.2 Objetivos

El presente trabajo tiene como principal objetivo la integración de un modelo de energía sobre una red LoRaWAN bajo la plataforma de simulación NS-3, para llevar a cabo un análisis de la eficiencia energética de los dispositivos de la red que funcionan bajo sistemas de alimentación autónoma, planteando para ello escenarios reales alineados con aplicaciones de interés.

Para alcanzar tal objetivo se establecen una serie de puntos a llevar a cabo:

- Caracterización de las redes baja potencia y área amplia (*Low Power Wide Area Network*, LPWAN), y tecnologías presentes en el mercado IoT actual.
- Análisis y estudio del *stack* de comunicación y arquitectura de LoRaWAN.
- Descripción, configuración y manejo del entorno de simulación de red NS-3 basado en eventos discretos para la simulación de redes como LoRaWAN.
- Análisis del modelo de red LoRaWAN implementado en [4], desarrollo que toma como punto de partida el presente trabajo.
- Diseño de un modelo energético que caracterice las fuentes de energía y el consumo energético de los dispositivos finales (ED) en una red LoRaWAN. Implementación del modelo teórico, diseño de la arquitectura de software y codificación, establecimiento de métricas, test y validación.
- Planteamiento de los escenarios de aplicación del modelo y ejecución de su simulación. Análisis de datos y propuestas de mejora para optimizar la eficiencia energética de los dispositivos en LoRaWAN.

1.3 Enfoque y método seguido

Una vez definidos los objetivos, y establecida la línea temporal de desarrollo del proyecto, el trabajo ha seguido una línea continuista que ha tomado como referencia de partida el desarrollo presente en [4], un entorno ya validado, reduciendo considerablemente el tiempo de desarrollo que supondría el diseño desde el inicio de una red LoRaWAN en un entorno de simulación como NS-3.

De este modo, el presente trabajo se centra exclusivamente en la integración del modelo de energía de los dispositivos de la red para su posterior análisis, facilitando con ello el proceso de implementación practica, y fomentando el desarrollo colaborativo en el cual se basa la plataforma NS-3. La elección de un entorno simulado facilitada por otro lado el análisis en redes de alta densidad que resultaría complejo llevar a cabo en escenarios reales debido a limitaciones logísticas.

El trabajo ha seguido una evolución teórico-práctica, comenzando con una etapa inicial de análisis "de arriba a abajo", caracterizando las redes LPWAN en el contexto de IoT, analizando la arquitectura y *stack* de LoRaWAN, así como su modelo y simulación implementado en [4], para finalmente implementar un modelo teórico de energía, definir un diseño de software, y llevar a cabo su codificación e integración en la plataforma NS-3 para su posterior análisis.

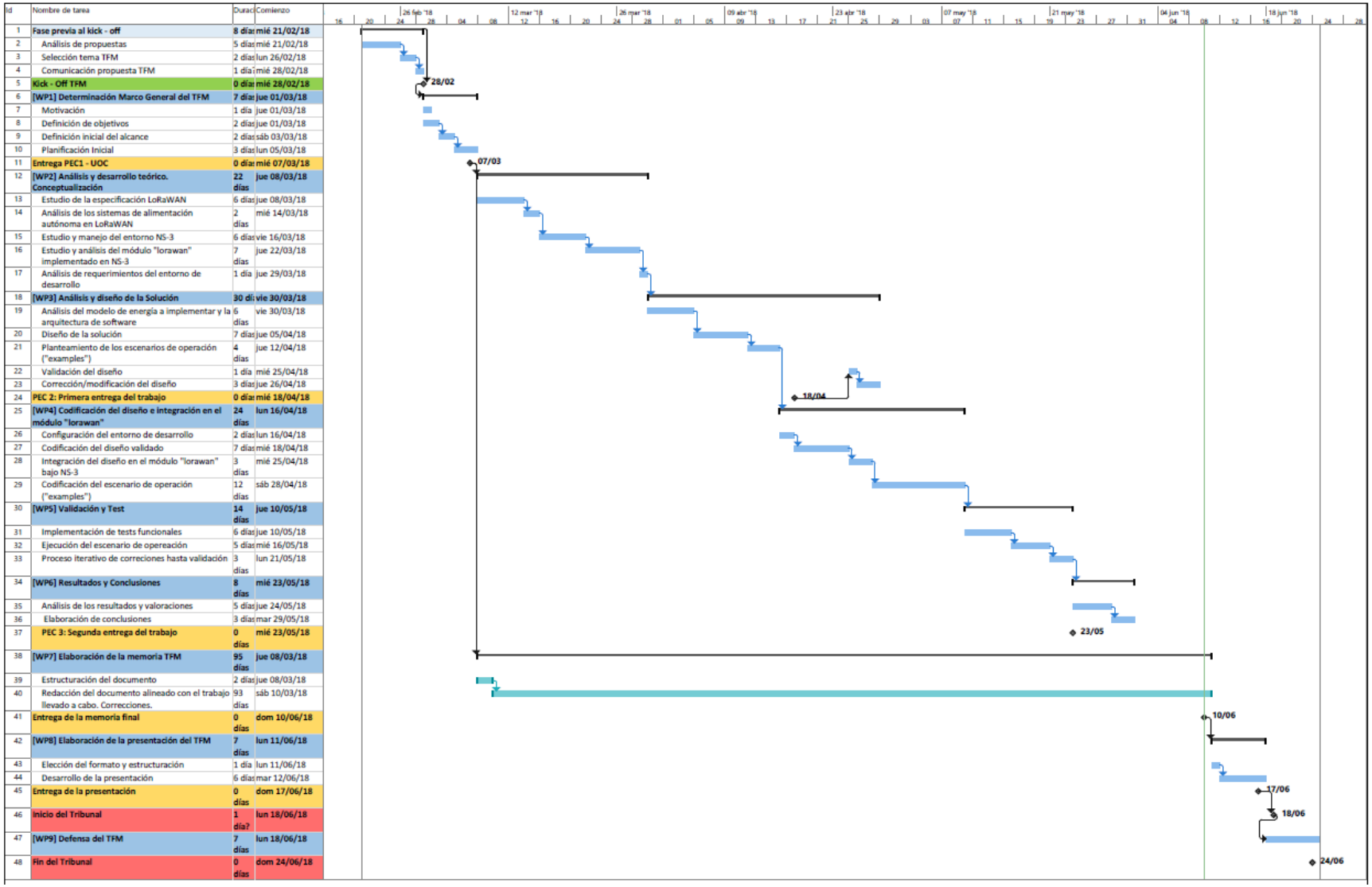
Para alcanzar los objetivos planteados, se han llevado a cabo sendas etapas de planificación y ejecución y control de acuerdo al modelo de referencia PMBOK, estableciéndose una serie de paquetes de trabajos (WP) bien definidos, así como una serie de puntos de seguimiento, con el objetivo de alcanzar los estándares de calidad esperados en un trabajo de estas características, de acuerdo al tiempo y alcance establecidos.

1.4 Planificación del Trabajo

La planificación del presente trabajo se ha estructurado de acuerdo a una serie de paquetes de trabajo y ha seguido una evolución alineada con los hitos establecidos en el plan docente del TFM.

Entregas parciales	Paquetes de trabajo	Fecha
Propuesta TFM	N/A – Fase previa al *kick-off*	**28/02/2018**
PEC1: Planificación del trabajo	**[WP1]** Determinación del Marco General del TFM.	**07/03/2018**
PEC2: Primera entrega (60% trabajo técnico)	**[WP2]** Análisis y desarrollo teórico. Conceptualización. **[WP3]** Análisis y diseño de la solución.	**18/04/2018**
PEC3: Segunda entrega (100% trabajo técnico)	**[WP4]** Codificación del diseño e integración. **[WP5]** Validación y ejecución de escenarios.	**23/05/2018**
Entrega de la memoria final	**[WP6]** Valoración y conclusiones. **[WP7]** Elaboración de la memoria.	**10/06/2018**
Entrega de la presentación	**[WP8]** Elaboración de la presentación del TFM.	**17/06/2018**
Inicio del tribunal	**[WP9]** Defensa del TFM.	**18/06/2018**
Final del tribunal		**24/06/2018**

Tabla 1. Correspondencia entre WPs y entregas parciales.



Id	Nombre de tarea	Duraci	Comienzo
1	**Fase previa al kick - off**	**8 días**	**mié 21/02/18**
2	Análisis de propuestas	5 días	mié 21/02/18
3	Selección tema TFM	2 días	lun 26/02/18
4	Comunicación propuesta TFM	1 día	mié 28/02/18
5	**Kick - Off TFM**	**0 días**	**mié 28/02/18**
6	**[WP1] Determinación Marco General del TFM**	**7 días**	**jue 01/03/18**
7	Motivación	1 día	jue 01/03/18
8	Definición de objetivos	2 días	jue 01/03/18
9	Definición inicial del alcance	2 días	sáb 03/03/18
10	Planificación Inicial	3 días	lun 05/03/18
11	**Entrega PEC1 - UOC**	**0 días**	**mié 07/03/18**
12	**[WP2] Análisis y desarrollo teórico. Conceptualización**	**22 días**	**jue 08/03/18**
13	Estudio de la especificación LoRaWAN	6 días	jue 08/03/18
14	Análisis de los sistemas de alimentación autónoma en LoRaWAN	2 días	mié 14/03/18
15	Estudio y manejo del entorno NS-3	6 días	vie 16/03/18
16	Estudio y análisis del módulo "lorawan" implementado en NS-3	7 días	jue 22/03/18
17	Análisis de requerimientos del entorno de desarrollo	1 día	jue 29/03/18
18	**[WP3] Análisis y diseño de la Solución**	**30 días**	**vie 30/03/18**
19	Análisis del modelo de energía a implementar y la arquitectura de software	6 días	vie 30/03/18
20	Diseño de la solución	7 días	jue 05/04/18
21	Planteamiento de los escenarios de operación ("examples")	4 días	jue 12/04/18
22	Validación del diseño	1 día	mié 25/04/18
23	Corrección/modificación del diseño	3 días	jue 26/04/18
24	**PEC 2: Primera entrega del trabajo**	**0 días**	**mié 18/04/18**
25	**[WP4] Codificación del diseño e integración en el módulo "lorawan"**	**24 días**	**lun 16/04/18**
26	Configuración del entorno de desarrollo	2 días	lun 16/04/18
27	Codificación del diseño validado	7 días	mié 18/04/18
28	Integración del diseño en el módulo "lorawan" bajo NS-3	3 días	mié 25/04/18
29	Codificación del escenario de operación ("examples")	12 días	sáb 28/04/18
30	**[WP5] Validación y Test**	**14 días**	**jue 10/05/18**
31	Implementación de tests funcionales	6 días	jue 10/05/18
32	Ejecución del escenario de opereación	5 días	mié 16/05/18
33	Proceso iterativo de correciones hasta validación	3 días	lun 21/05/18
34	**[WP6] Resultados y Conclusiones**	**8 días**	**mié 23/05/18**
35	Análisis de los resultados y valoraciones	5 días	jue 24/05/18
36	Elaboración de conclusiones	3 días	mar 29/05/18
37	**PEC 3: Segunda entrega del trabajo**	**0 días**	**mié 23/05/18**
38	**[WP7] Elaboración de la memoria TFM**	**95 días**	**jue 08/03/18**
39	Estructuración del documento	2 días	jue 08/03/18
40	Redacción del documento alineado con el trabajo llevado a cabo. Correcciones.	93 días	sáb 10/03/18
41	**Entrega de la memoria final**	**0 días**	**dom 10/06/18**
42	**[WP8] Elaboración de la presentación del TFM**	**7 días**	**lun 11/06/18**
43	Elección del formato y estructuración	1 día	lun 11/06/18
44	Desarrollo de la presentación	6 días	mar 12/06/18
45	**Entrega de la presentación**	**0 días**	**dom 17/06/18**
46	**Inicio del Tribunal**	**1 día?**	**lun 18/06/18**
47	**[WP9] Defensa del TFM**	**7 días**	**lun 18/06/18**
48	**Fin del Tribunal**	**0 días**	**dom 24/06/18**

Figura 1. Planificación detallada.

Durante el desarrollo del trabajo se ha llevado a cabo un seguimiento periódico de las actividades que forman parte de los diferentes paquetes de trabajo (WP) establecidos en la planificación:

- **[WP1] Determinación del Marco General del TFM**. Tras recibir el primer informe de evaluación por parte de la tutorización, se ha dado especial importancia a la necesidad del estudio del estado del arte en la fase de desarrollo teórico [WP2] y en el diseño de la solución [WP3], así como a establecer unas métricas bien definidas para validar el modelo implementado en [WP5], y en definir escenarios de operación que se ajusten a aplicaciones de interés.

- **[WP2] Análisis y desarrollo teórico. Conceptualización**. Las fases de estudio de especificación [WP2.1], de análisis de sistemas de alimentación autónoma [WP2.2] se han llevado a cabo en el tiempo previsto, sin embargo, la fase de estudio y manejo del entorno de NS-3 [WP2.3] ha supuesto pequeño tiempo de demora debido a la complejidad del entorno NS-3. Tras analizar entorno, las fases de estudio del módulo LoraWAN [WP2.4], y la de análisis de requerimientos [WP2.5] se han completado en el tiempo previsto.

- **[WP3] Análisis y diseño de la solución:** El análisis del modelo y la arquitectura de software [WP3.1] se ha completado en el tiempo considerado, siendo su carga más ligera de lo previsto debido a la disponibilidad de modelos de energía de referencia en NS-3 tales como el presente en el módulo *Wifi*. Sin embargo, en la fase de diseño de la solución [WP3.2], ha habido ciertas dificultades para modelar el consumo de corriente de los dispositivos dada la propiedad de la tecnología LoRa, definiéndose finalmente un modelo de corriente basado en interpolación. En cuanto al planteamiento de los escenarios de operación [WP3.3], ha habido un retraso considerable, debido a la no disponibilidad en el repositorio [4] de modelos de propagación adecuados para la generación de escenarios simulados, surgiendo la necesidad de analizar otros modelos de propagación apropiados presentes en NS-3. Por otro lado, la validación de los modelos teóricos de fuente de energía y consumo de corriente en modo de transmisión [WP3.4] se ha llevado a cabo tras la entrega parcial PEC2, no de forma previa a la misma como se había programado inicialmente, debida a la fase todavía prematura del trabajo.

- **[WP4] Codificación del diseño e integración:** Las fases de configuración del entorno [WP4.1], así como la de codificación [WP4.2] e integración [WP4.3], se han llevado a cabo en el tiempo previsto. Sin embargo, el proceso de codificación de escenarios [WP4.4] ha experimentado una fuerte demora como consecuencia de la necesidad de integración de los modelos de propagación en los escenarios simulados, así como del establecimiento de mecanismos de posicionamiento de los nodos en la simulación mediante el uso de modelos estándar de NS-3.

- **[WP5] Validación y test:** En las fases de implementación de tests [WP5.1], ejecución de escenarios [WP5.2] y correcciones [WP5.3] ha habido también un considerable tiempo de demora debido a la necesidad de implementación de herramientas de captura de datos para realizar análisis posteriores, actividad que no se había contemplado en el plan inicial.

- **[WP6] Resultados y conclusiones:** La fase análisis de resultados [WP6.1] y elaboración de conclusiones [WP6.2] se han llevado a cabo tras la entrega parcial PEC3, y no previamente, tal y como se había previsto debido a las incidencias acontecidas en las fases [WP4] y [WP5].

- **[WP7] Elaboración de la memoria del TFM:** La redacción del documento de la memoria ha sido llevada a cabo progresivamente durante las diferentes entregas parciales establecidas en la tabla 1, y no tras la entrega parcial PEC3, como se estableció inicialmente, para permitir una corrección progresiva del documento. A su vez, debido a las incidencias durante las fases [WP4] y [WP5], las correcciones de la entrega parcial PEC2 no han podido llevarse a cabo previo a la entrega parcial PEC3, siendo pospuestas para la entrega final de la memoria.

1.5 Breve sumario de productos obtenidos

El presente TFM contiene los siguientes *outputs* o entregables:

- Análisis de la especificación LoRaWAN y del modelo de software implementado en [4]
- Diseño del modelo teórico de energía y consumo en los dispositivos finales (ED) de una red LoRaWAN.
- Modelado de la arquitectura de software implementada (diagramas UML).
- Codificación en C++ del modelo energético e integración en el módulo implementado en [4]
- Codificación en C++ de los escenarios planteados en la simulación.
- Captura, filtro y análisis de los datos capturados en las simulaciones.

1.6 Breve descripción de los otros capítulos de la memoria

El presente documento presenta la siguiente estructuración:

- **Capítulo 2. Redes LPWAN y LoRaWAN.** Características de las redes LPWAN y tecnologías con mayor nicho de mercado en la actualidad. Descripción de la arquitectura LoRaWAN, y las capas físicas y de enlace del *stack.*
- **Capítulo 3. Entorno de simulación de red NS-3:** Conceptualización, estructura y terminología del entorno de simulación NS-3. Herramientas principales para operar en el entorno.

- **Capítulo 4. Simulación de LoRaWAN.** Análisis del modelo de red LoRaWAN implementado en [4]. Modelado de software de los principales componentes implicados en el proceso de integración del modelo de Energía.
- **Capítulo 5. Integración de un modelo de energía.** Alcance de la solución implementada, estado del arte. Diseño, codificación, test, validación del modelo, descripción de escenarios.
- **Capítulo 6. Despliegue y resultados.** Análisis de los datos obtenidos de la simulación y valoración global respecto a los resultados esperados.
- **Capítulo 7. Conclusiones.** Lecciones aprendidas durante el desarrollo del trabajo, análisis sobre el grado de consecución de objetivos de acuerdo al alcance inicial, definición de líneas futuras.
- **Capítulo 8. Glosario.** Definición de términos y acrónimos relevantes utilizados en el presente documento.
- **Capítulo 9. Bibliografía.** Referencias bibliográficas utilizadas en el presente documento.

2. Redes LPWAN y LoRaWAN.

2.1 Introducción

Con la irrupción y la creciente evolución del paradigma *Internet of Things (*IoT*)* en los últimos años, una serie de nuevas tecnologías han ido emergiendo con el objetivo de dar respuesta a las necesidades técnicas inherentes a este nuevo concepto de plena conectividad o interconexión del "todo" que las tecnologías tradicionales no han podido satisfacer. Los nuevos protocolos y estándares de comunicación deben dar respuesta a las diferentes particularidades asociadas a esta nueva concepción de interconectividad:

- Bajo consumo de energía: Dado el amplio volumen de dispositivos presentes en las diferentes topologías de red en IoT, un factor importante a tener en cuenta es la autonomía de los dispositivos que operan bajo sistemas de alimentación de batería, con el fin de reducir los costes de operación y de mantenimiento de la red. Lo que implica, a su vez, que las capacidades computacionales de los dispositivos sean bajas y los protocolos deban adaptar los mecanismos y esquemas de modulación a tal fin.
- Movilidad en las comunicaciones: Multitud de escenarios en IoT poseen exigentes requerimientos de movilidad, factor que las nuevas tecnologías han de contemplar para minimizar los efectos adversos subyacentes al dinamismo de los objetos que operan en ellos.
- Dispositivos de bajo coste: Para garantizar la viabilidad de las tecnologías IoT y asegurar un nicho en el mercado, el coste de los dispositivos es un elemento de crucial importancia dado el orden de magnitud de dispositivos que pueden operar bajo una misma red.
- Seguridad: Los grandes volúmenes de datos presentes las redes IoT, así como los nuevos escenarios que plantea el nuevo paradigma demandan robustos mecanismos de seguridad a diferentes niveles de la pila de protocolo (*stack*), así como de la topología de red.
- Flexibilidad, adaptabilidad y escalabilidad: Debido al gran dinamismo en la operatividad de las redes IoT, los nuevos estándares y protocolos deben contemplar un alto grado de flexibilidad, adaptabilidad, así como escalabilidad para garantizar un funcionamiento eficiente de la red y reducir las operaciones de mantenimiento en la misma.

Tal y como puede comprobarse en la literatura actual, existen diferentes tecnologías con un amplio nicho en el mercado IoT que dan respuesta a estas necesidades técnicas, tales como IEEE 802.15.4 (6LoWPAN, Z-Wave, Thread), IEEE P802.1ah, Bluetooth/LE, NB-*IoT*, C*IoT*, etc. El presente trabajo se centra en LoRaWAN, una prominente tecnología, todavía en desarrollo, que pretende competir con las tecnologías IoT actuales en el contexto de las redes de baja potencia y área amplia - *Low-Power Wide Area Network* (LPWAN).

2.2 Redes de baja potencia y área amplia (LPWAN)

En el contexto de IoT, las redes de baja potencia y área amplia (LPWAN) surgen para dar respuesta a la necesidad de uso de conexiones de largo alcance con un reducido consumo de potencia, y permitir su uso en escenarios IoT como *Smart Cities*, *Smart Agriculture, Smart Transportation* ampliando así el nicho de mercado IoT ya existente mediante el uso de tecnologías como *Bluetooth Low Energy (BLE)*, *6LowPAN*, *Zigbee*, *Thread*, *Z-Wave*, etc., asociadas a redes WPAN/LPLAN que operan bajo escenarios como *Smart Homes, Wereables,* etc. donde los requerimientos de alcance son considerablemente menores. Varias son las características técnicas [21] que definen a las redes LPWAN y que garantizan los requisitos anteriores asociados a las tecnologías IoT.

La topología que predomina en las redes de corto alcance WPAN/LPAN para extender la cobertura de red es la topología de malla, en la cual existen múltiples caminos disponibles en el establecimiento de enlaces entre nodos, evaluados por los algoritmos de enrutamiento. Mediante el uso de esta topología se otorga una mayor robustez a la red ante posibles caídas de nodos intermedios, pero implica una mayor de velocidad de transmisión de datos, y por ende un mayor consumo, ante la necesidad de compensar los retrasos inherentes a la comunicación multi-salto. En cambio, la topología de red predominante en redes LPWAN, donde los dispositivos finales (ED) están caracterizados por una alta sensibilidad en lugar de altas tasas de transmisión, es la topología en estrella, en la cual los ED están directamente conectados con un dispositivo central, que opera como transductor de protocolo desde los ED hasta otros puntos remotos haciendo uso de protocolos IP estándares de amplio alcance (WAN) y siguiendo estrategias de *Fog Computing* o *Edge Computing* [19] según sean las necesidades de la aplicación o servicio. Las características de bajo consumo de las redes en estrella permiten el uso de dispositivos a través de sistemas de alimentación autónoma explotando al máximo el tiempo de operación de los ED.

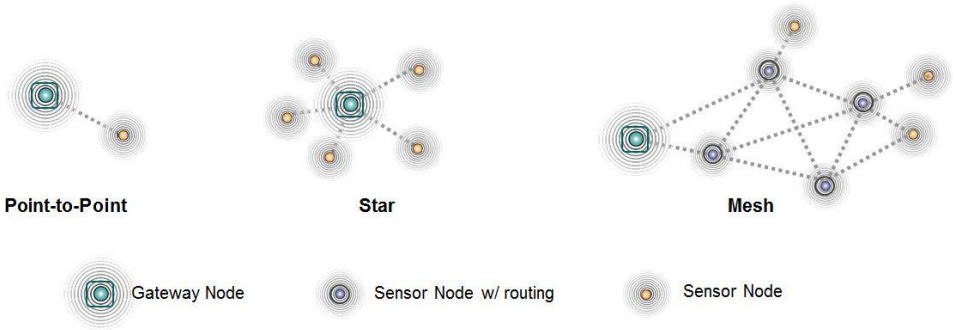


Figura 2. Topologías de red: Punto a punto, estrella y malla [20].

En las redes LPWAN cabe distinguir dos grandes grupos de técnicas de modulación: Modulación de banda ultra estrecha (*Ultra Narrow Band,* UNB) y modulación de espectro ensanchado (*Spread Spectrum,* SS). Las redes LPWAN que hacen uso de modulaciones UNB operan en de una banda frecuencial en torno a 100 Hz, y suelen utilizar técnicas de modulación que son variaciones de otras técnicas ya ampliamente usadas en el ámbito de las comunicaciones inalámbricas, tales como la como la modulación por desplazamiento de amplitud (*Amplitude-Shift Keying,* ASK*),* modulación por desplazamiento de frecuencia

(*Frequency-Shift Keying,* FSK*)* y modulación por desplazamiento de fase (*Phase-Shift Keying)* PSK. Una mayor complejidad en la técnica implica un mayor número de bits transmitidos por símbolo. Por ejemplo, BPSK, dispone dos símbolos en su diagrama de constelación, con un bit de información cada uno y con un salto de fase de 180º, obteniendo una tasa de $1bit/símbolo$. En cambio, QPSK dispone de cuatro símbolos en su diagrama de constelación, con dos bits de información por símbolo y un salto de fase de 90º, confiriéndole una tasa de $2bit/símbolo$, a consta de una reducción de la inmunidad frente al ruido dada la menor separación entre símbolos. La técnica de modulación de espectro ensanchado está basada en el "ensanchamiento" de la señal a transmitir en una banda frecuencial mucho más amplia que la mínima requerida para transmitir la información de la señal original. La modulación SS contempla diferentes técnicas para llevar a cabo el ensanchamiento frecuencial de la señal, entre las que cabe destacar los sistemas de secuencia directa (*Direct Sequence Spread Spectrum,* DSSS), en los cuales la dispersión de energía se realiza mediante una pre-modulación de la señal original utilizando una secuencia pseudoaleatoria de gran velocidad que incrementa la banda frecuencial de la señal producto; los sistemas de salto de frecuencia (*Frequency Hopping Spread Spectrum,* FHSS*)* donde el valor de la frecuencia de la señal portadora cambia de acuerdo a una secuencia pseudoaleatoria; los sistemas de salto temporal, basados en la variación aleatoria del periodo y el ciclo de trabajo de la señal portadora, habitualmente utilizados en conjunto con sistemas de salto frecuencial; los sistemas de frecuencia modulada pulsada (*chirping*) que utilizan un pulso para realizar un barrido frecuencial acotado, denominado *chirp*, expandiendo la señal espectral; así como otros sistemas híbridos que combinan varios de los métodos anteriores para beneficiarse de las ventajas que aporta cada uno.

La mayoría de las redes LPWAN operan en bandas inferiores a 1Ghz, lo que proporciona un gran grado de robustez y confiabilidad sin unos altos requisitos en potencia. En comparación con la banda de 2,4GHz [21] las señales de frecuencia inferiores experimentan menos atenuación y desvanecimiento en escenarios con multi-camino y obstáculos de gran densidad. A su vez, la banda sub-Ghz está menos congestionada, ya que los dispositivos no deben convivir con tecnologías como Wi-Fi, Blueetooth, ZigBee, etc. como si ocurre en la banda de 2,4GHz.

Los mecanismos de acceso al medio utilizados en LPWAN son más ligeros computacionalmente en comparación con los utilizados en redes celulares o redes inalámbricas de corto alcance. La complejidad tecnológica en redes LPWAN no se lleva a cabo en los dispositivos finales (*ED*), sino en las estaciones base. Varias tecnologías como Sigfox y LoRaWAN hacen uso de protocolos basados en ALOHA, un protocolo MAC de acceso aleatorio al medio en el que los ED transmiten si realizar ninguna detección de portadora. La simplicidad de ALOHA mantiene el diseño del transceptor RF simple y de bajo coste. Otras tecnologías como INGENU y NB-IoT consideran otros protocolos MAC basados en TDMA para asignar recursos de radio de manera más eficiente, pero a costa de una mayor complejidad y coste.

Los dispositivos finales (*ED*) aplican técnicas de *Duty Cycling* para maximizar su consumo de corriente, desactivando componentes de hardware con altos

requerimientos de consumo, tales como los transceptores RF (*RF-transceivers*) cuando no se requiere de transmisiones en la red. En algunos estándares, como LoRaWAN, se establecen diferentes técnicas de *Duty Cycling*, clasificando a los dispositivos por clases en función del modo de operación adoptado.

En referencia al coste de la red, en LPWAN se hace un esfuerzo por reducir la complejidad del hardware de los dispositivos finales. Los transceptores necesitan procesar formas de onda menos complejas lo que supone a su vez una menor velocidad de datos y menor tamaño de memorias internas. A su vez, la mayoría de las tecnologías hacen uso de bandas exentas de licencia, incluida la banda industrial, científica y médica (ISM) o los espacio en blanco de TV.

El soporte para una cantidad masiva de ED se lleva a cabo a través de la aplicación de técnicas de diversidad en diferentes planos: canal, tiempo y espacio. Las tecnologías LPWAN emplean comunicación multicanal y técnicas MIMO en los dispositivos centrales para paralelizar las transmisiones con los ED. La adaptación de los esquemas de modulación, la selección de mejores canales, el control adaptativo de la potencia de transmisión requiere a su vez una monitorización eficiente de las cualidades de los enlaces, así como la coordinación entre los dispositivos finales y la red.

En la actualidad existen varias tecnologías que han surgido recientemente en el contexto de las redes LWPAN [21] entre ellas cabe destacar Sigfox, Ingenu, Telensa y Lora por su relevancia actual. En la Tabla 2, pueden observarse las especificaciones técnicas de cada uno de ellos.

- **Sigfox:** Sigfox ofrece una solución de conectividad de extremo a extremo basada en una tecnología propietaria. Los ED se conectan a las estaciones base utilizando modulación BPSK en una banda UNB de 100Hz, consiguiendo un bajo nivel de ruido, una alta sensibilidad de recepción y un consumo de energía ultra bajo, pero una tasa de bits de 100bps, cualitativamente menor que otras tecnologías. Opera en la banda frecuencial 868.180 – 868.20 MHz en Europa, dividida en 400 canales (360 útiles) de 100Hz.
- **LoRaWAN:** LoRaWAN hace uso de una tecnología de capa física propietaria, denominada LoRa, que aplica una modulación basada en *Spread Spectrum* en la banda sub-Ghz ISM, que hace uso de diferentes factores de dispersión (*Spread Factor*, SF), caracterizada por una alta inmunidad al ruido e interferencia. LoRa combina mecanismos de detección de errores (FEC) con la técnica de espectro ensanchado para aumentar la sensibilidad del receptor. Su tasa de bits varía entre 300 bps y 37,5 kbps dependiendo del SF y del ancho de banda del canal. Opera en las bandas de 433Mhz y 868MHz en Europa. Los mensajes transmitidos por los ED son recibidos por más de una estación base, constituyendo lo que se denomina una topología “estrella de estrellas”, consiguiendo con ello diversidad de recepción, lo que le permite además aplicar técnicas de localización de los ED a través de métodos basados en la diferencia de tiempo de llegada (TDOA).
- **Ingenu:** Ingenu no se basa en las propiedades de propagación de la banda sub-Ghz, como otras tecnologías, sino que opera en la banda ISM

de 2,4 Ghz donde las regulaciones sobre el uso del espectro están sujetas a menos restricciones, confiriéndole un mayor rendimiento y capacidad. Utiliza un esquema patentado de acceso al medio denominado Acceso Múltiple por Fases Aleatorias (*Random Phase Multiple Access*, RPMA) mediante el cual varios transmisores pueden compartir un único intervalo de tiempo. A través de esta técnica se reduce la superposición entre señales transmitidas aumentado la relación de señal a interferencia para cada enlace individual. En el enlace descendente (DL) las estaciones base hacen uso de modulación CDMA. Con RPMA se consiguen sensibilidades de recepción de hasta -142dBm

- **Telensa:** Telensa proporciona soluciones de extremo a extremo que incorporan *stacks* de comunicación diseñados para ser integrado con software de terceros. Hace uso de una técnica de modulación UNB patentada que opera en la banda ISM sub-Ghz sin licencia a bajas velocidades de datos que oscilan alrededor de 62.5 bps en el enlace ascendente y 500 bps en el enlace descendente.

	Sigfox	**LoRaWAN**	**Ingenu**	**Telensa**
Modulación	UNB DBPSK(UL), GFSK(DL)	CSS	RPMA-DSSS(UL), CDMA(DL)	UNB 2-FSK
Banda frecuencial	SUB-GHz ISM: EU (868 MHz), US(902MHz)	SUB-GHz ISM: EU (433MHz, 868Mhz), US(915MHz), Asia(430MHz)	ISM 2,4 GHz	SUB-GHz ISM: EU(868Mhz), US(915MHz), Asia(430MHz)
Tasa de bits	100 bps(UL), 600 bps(DL)	0.3-37.5 kbps(LoRa), 50 kbps(FSK)	78kbps(UL), 19.5 kbps(DL)	62.5(UL), 500 bps (DL)
Alcance	10 km (urbano), 50 km (rural)	5 km (urbano), 15 km(rural)	15 km (urbano)	1km (urbano)
Número de canales/ Señales ortogonales	360 canales	10 en EU, 64+8(UL) – 8DL en US	40 canales de 10MHz, hasta 1200 señales por canal	Múltiples canales
Simetría de enlace	No	Sí	No	No
FEC	No	Sí	Sí	Sí
MAC	ALOHA	ALOHA	CDMA	-
Topología	estrella	Estrella de estrellas	Estrella, árbol	estrella
ADP	No	Sí	Sí	No
Longitud de datos útiles (*Payload)*	12 B (UL), 8 B (DL)	Hasta 250B	10KB	-
Handhover	EDs no se suscriben a una única estación base	EDs no se suscriben a una única estación base	Sí	No
Autenticación y encriptación	No	AES 128 b	16B hash, AES 256 b	-
Actualizaciones OTA	No	Sí	Sí	Sí
Soporte SLA	No	No	No	No
Localización	No	Sí	No	No

Tabla 2. Tabla comparativa entre tecnologías LPWAN. Sigfox, LoraWAN, Ingenu, Telensa [21].

Pese a que todas las tecnologías contempladas en la tabla comparativa poseen unas altas prestaciones de alcance y a su vez dan respuesta a las nuevas necesidades técnicas descritas en 2.1, LoRaWAN merece una especial atención debido a su gran relación alcance/velocidad de datos, su operación en múltiples bandas frecuenciales, un tamaño de *payload* adecuado para una gran diversidad de aplicaciones y su gran popularidad en el mercado con respecto al resto de tecnologías, siendo por ello la tecnología en la cual se centra el presente trabajo.

2.3 LoRaWAN

LoRaWAN es una especificación de red LPWAN publicada por el grupo LoRa Alliance [24], que hace uso de la tecnología propietaria LoRa correspondiente a la capa física del *stack*, con el fin de explotar la arquitectura celular adoptada típicamente en soluciones IoT. La norma únicamente cubre la capa de enlace situada por encima de la tecnología propietaria, pero no proporciona especificaciones sobre capas superiores. En los siguientes apartados se lleva a analizan los aspectos más relevantes del *stack* presentes en la literatura, [22] y [23]que ayudarán a entender la solución adoptada en el presente trabajo.

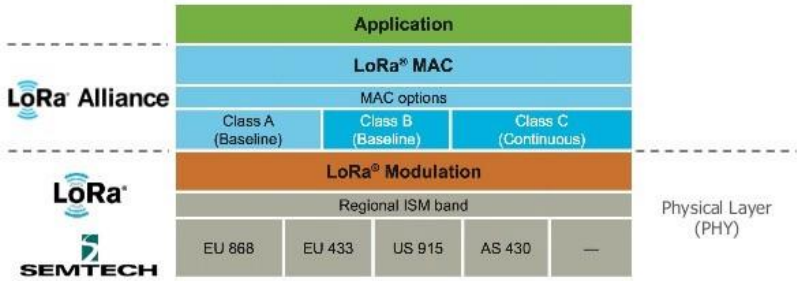


Figura 3. Stack *de comunicación de LoRaWAN* [48] .

2.3.1 Arquitectura

La arquitectura de LoRaWAN está formada por una serie de nodos que operan como dispositivos finales (*End Devices*, ED), un servidor de red (*Network Server, NS*) que realiza la captura y análisis de datos enviados desde los ED, y puertas de enlace (*Gateways,* GW*)* que realizan el renvío de paquetes entre los nodos finales y el servidor de red, conformando lo que se denomina una topología de "estrella de estrellas".

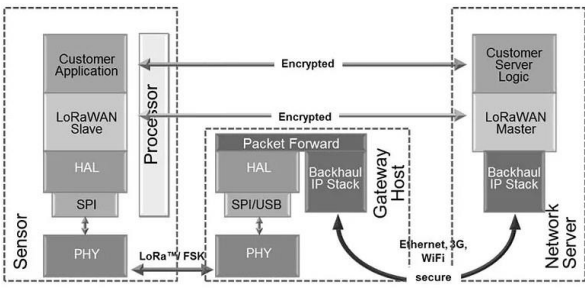


Figura 4. Estructura *Peer-to-Peer* [49].

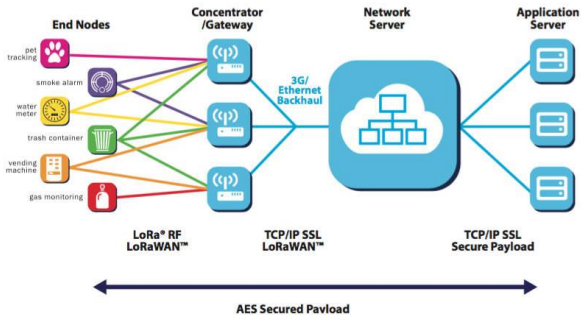


Figura 5 Arquitectura LoRaWAN [25].

LoRaWAN estable una serie de clases para los ED en función de su modalidad de consumo:

- **Clase A:** Disponen estrictos requerimientos de consumo y eficiencia energética. Soportan la funcionalidad y mecanismos básicos de LoRaWAN. Permiten comunicaciones bidireccionales con mecanismos de acceso al medio basados en ALOHA no ranurado, de acuerdo a las

necesidades de transmisión. Las transmisiones de enlace descendente son llevadas a cabo durante intervalos de tiempo selectivos acotados por una serie de ventanas de recepción en los ED.

- **Clase B:** Poseen de ventanas de recepción adicionales a las especificadas por la clase A que pueden ser programadas a través de un sistema de balizas. Esta clase está asociada a ED que son contralados de forma remota.
- **Clase C:** Disponen de una ventana de recepción permanente. Esta clase es idónea para ED que son controlados de forma remota y disponen de restricciones de tiempo respuesta.

2.3.2 Capa física

La capa física de Lora hace uso de un esquema de modulación basada *Chirp Spread Spectrum (*CCS*),* usada ampliamente en el ámbito militar y en comunicaciones seguras debido a su robustez y bajos requerimientos de potencia*.* En CCS cada símbolo está representado a través de una señal sinusoidal cuya frecuencia cambia cíclicamente dentro de un ancho de banda acotado, B, alrededor de una frecuencia central f_c. A partir de un valor inicial, la frecuencia va incrementándose hasta llegar a un máximo de $f_c + \frac{B}{2}$, momento en el que salta a la cota mínima $f_c - \frac{B}{2}$ para seguir aumentando hasta repetir el ciclo de nuevo.

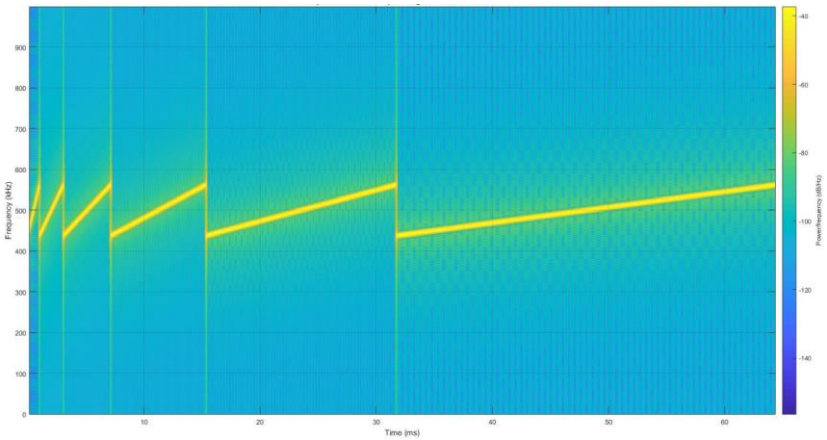


Figura 6. Comparativa entre diferentes valores de SF en LoRaWAN [50].

El número de valores de frecuencia iniciales viene dado por 2^{SF}, siendo SF (*Spread Factor)* el número de bits por símbolo transmitido, que puede variar entre los valores [7,12]. A su vez, LoRa, implementa un mecanismo de corrección de errores aplicando diferentes valores de CR *(Code Rate)* entre $[\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}]$ para ajustar el valor del FEC (*Forward Error Correction*) deseado. Los valores de B, CR y SF definen la tasa de bits R_b de LoRa en un enlace punto a punto.

$$T_s = \frac{2^{SF}}{B} \qquad (2.1) \qquad\qquad R_b = \frac{SF}{T_s} CR = \frac{SF \cdot B}{2^{SF}} CR \qquad (2.2)$$

T_S: *Duración del símbolo*

Cuanto mayor es el valor de SF menor es la tasa de bits R_b, pero mayor es la sensibilidad de los dispositivos. En la tabla 3 puede observarse la correspondencia entre diferentes valores de SF, CR, B para la banda frecuencial ISM utilizada en Europa de 868 MHz.

DR	SF	Ancho de banda de canal [kHz]	CR	Tasa de bits [bps]	Sensibilidad [dBm]
0	12	125	4/6	250	-137
1	11	125	4/6	440	-136
2	10	125	4/5	980	-134
3	9	125	4/5	1760	-131
4	8	125	4/5	3125	-128
5	7	125	4/5	5470	-125
6	7	250	4/5	11000	-122

Tabla 3. Correspondencia entre SF, DR, CR, sensibilidad, etc. [22].

Una característica a tener en cuenta en la modulación LoRa es que un dispositivo puede recibir dos transmisiones solapadas que posean diferente SF bajo el mismo canal. Incluso en caso de dos transmisiones simultáneas con el mismo SF, un dispositivo puede realizar la recepción correctamente de la señal con mayor potencia, mientras el diferencial entre ellas sea mayor de 3dB.

La trama de la capa física en Lora está formada por un preámbulo, que facilita la sincronización y define el esquema de modulación definiendo un SF determinado, cuya duración suele ser de 12.25 T_s, seguido por la cabecera PHY y la cabecera CRC que forman un total de 20 bits codificados con un código de 4/8, mientras que el resto de la trama es codificada de acuerdo al ratio especificado en la cabecera PHY, que a su vez contiene también información como la longitud de los datos útiles (*payload*) y si se aplica o no un mecanismo CRC de 16 bits en el caso del sentido ascendente del enlace.

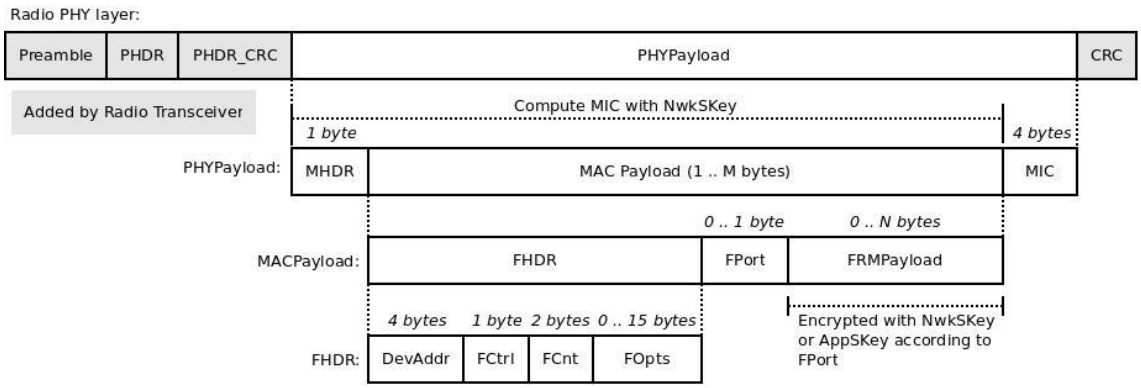


Figura 7. Formato de una trama en LoRaWAN [51].

El número de símbolos *n* requeridos para transmitir la cabecera y los datos útiles de la capa física, viene dado por la ecuación 2.3:

$$n = 8 + max\left(\left[\frac{8PL - 4SF + 28 + 1[\,if\ CRC]16}{CR(SF - 2DE)}\right], 0\right) \qquad (2.3)$$

$PL: Payload\ [Bytes]$
$DE: Low\ Data\ Rate\ Enabled$

La duración de la trama, a su vez, viene dada por la ecuación 2.4:

$$T_{frame} = (n + 12.25)T_s \qquad (2.4)$$

La opción de optimización DE es un mecanismo que puede ser habilitado en LoRa para aumentar la robustez de las transmisiones frente a variaciones en frecuencia.

2.3.3 Capa de enlace

La frecuencia de operación de los diferentes canales viene determinada en LoRaWAN por la configuración establecida en los dispositivos que operan como *gateways* (GW*)*. El número de canales asignados dependen de las restricciones y regulaciones de la zona geográfica, así como de las opciones de red. Los denominados canales principales están reservados para transmisión de datos, mientras que un canal en sentido descendente está reservado para que los GW efectúen sus respuestas en la lógica de transmisión.

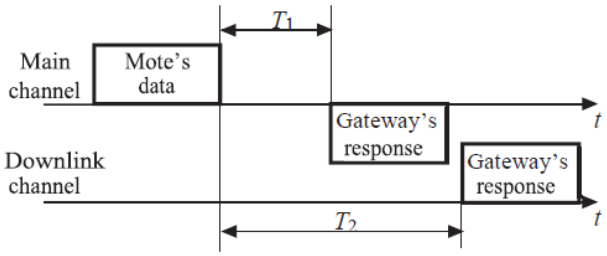


Figura 8. Acceso a un canal LoRa [24].

En el momento que un nodo ED de clase A tiene datos para transmitir, selecciona de forma aleatoria uno de los canales principales y realiza su transmisión siguiendo el protocolo de acceso al medio ALOHA no ranurado, sin sincronización ni sentido de portadora. Terminada la transmisión, el nodo abre dos ventanas de recepción, la primera en el canal utilizado para llevar a cabo la transmisión en sentido ascendente y la segunda en el canal de sentido descendente, durante el cual espera recibir la respuesta, en caso de que haya sido requerida en la cabecera MAC. La primera ventana de recepción permanece abierta durante T_1 segundos tras terminar la transmisión en sentido ascendente y la segunda un segundo más tarde durante T_2 segundos. Un GW sólo puede transmitir datos a los EDs como respuesta a estos durante las ventanas de recepción, si necesita transmitir datos de mayor tamaño que una trama, activa el bit FC para que el nodo ED inicie de nuevo el mecanismo anterior. Los dispositivos finales de clase B se sincronizan con el servidor de red, a través de un mecanismo de balizas (*beacons)* retransmitidas por los dispositivos GW. En este caso, los ED pueden recibir datos en ventanas de recepción de duración

específicas, independientemente del tráfico del enlace ascendente. Los dispositivos de clase C no tienen limitaciones temporales en las ventanas de recepción y pueden mantenerse a la escucha permanentemente.

LoRaWAN soporta envío de mensajes con y sin confirmación. Si un ED envía un mensaje que requiere de confirmación, el GW debería responderle activando el *flag* ACK durante las ventanas de recepción. Si es el GW el que envía un mensaje que requiere confirmación, el ED transmite la confirmación a su propia discreción. Si un nodo no recibe confirmación, retransmite el mensaje en un canal aleatorio en un periodo menor al ACK *Timeout*. La norma no especifica el valor del ACK *Timeout* pero se recomienda que tenga un valor aleatorio comprendido entre 1 y 3 segundos. La retransmisión del mensaje puede efectuarse de acuerdo a una tasa de bit menor, la norma especifica una tasa DR_r de asociada al intento *r de:*

$$DR_r = max\left(DR_0 - \frac{r}{2}, 0\right) \qquad (2.5)$$

El número recomendado de intentos es de 8, una vez excedido el paquete es descartado y se comunica a la capa superior que ha habido un error en la transmisión.

La trama de la capa de enlace está formada por una cabecera MAC de 1 byte que define la versión del protocolo y el tipo de mensaje (por ejemplo, si es una transmisión de datos o de control, si es una transmisión en sentido ascendente o descendente del enlace, si se requiere de ACK, etc.), seguida de una cabecera de trama que está constituida por los siguientes campos:

- Dirección del dispositivo: Los 8 primeros bits identifican la red y el resto de bits son asignados dinámicamente durante el acceso a la red para identificar el dispositivo suscrito.
- 1 byte de información de control: Información tal como el uso de la tasa de bits del GW en sentido de enlace ascendente, la necesidad de confirmación (*ACK*), sí el GW necesita transmitir más datos, etc.
- Número de secuencia.
- Opciones de trama: Comandos usados para cambiar la tasa o la potencia de transmisión, validar conexiones, etc.
- Puerto de trama: *Flag* que identifica el tipo de aplicación llevada a cabo. Si su valor es 0, la información útil (MAC-*payload*) contiene comandos MAC en lugar de datos.
- *MIC*: Firma digital del mensaje.

El tamaño total de la trama MAC es de $13 + FP + F_{Opts}$, donde FP es el tamaño del *payload* y F_{Opts} de los bits de opciones.

LoRaWAN permite el uso de un mecanismo adaptativo de tasa de bits ADR para controlar el tráfico de los dispositivos finales mediante el uso de comandos MAC. La velocidad de datos utilizada en el enlace descendente DR_{down} depende de la utilizada en el enlace ascendente DR_{up} y de un parámetro de red configurable, el $RX1DROffset$, cuyo valor varía en función de la zona geográfica

$$DR_{down} = \max\left(DR_{up} - RX1DROffset, 0\right) \qquad (2.6)$$

Por otro lado, debido a ciertas regulaciones existe una limitación del ciclo de trabajo de los dispositivos (*Duty Cycle*), por la cual un nodo no puede usar una banda frecuencial durante un cierto periodo de tiempo una vez que ha concluido su transmisión en esa banda. Definiendo el tiempo de transmisión como T_{OnAir}, el tiempo de reposo T_{Off}, viene impuesto por:

$$T_{Off} = \frac{T_{OnAir}}{DutyCycle} - T_{OnAir} \qquad (2.7)$$

Las bandas frecuenciales en las que opera LoRaWAN dependen de la zona geográfica, en las cuales se definen ciertos parámetros como el tiempo de preámbulo, las frecuencias centrales, los SF permitidos, el tamaño máximo de *payload*, las ventanas de recepción, etc. En la Tabla 4 pueden observarse las bandas frecuenciales asociadas a cada zona geográfica.

	Europa	**EEUU**	**China**	**Corea**	**Japón**	**India**
Banda [MHz]	867-869	902-928	470-510	920-925	920-925	865-867
Canales	10	64+8+8	N/A	N/A	N/A	N/A
BW Up [kHz]	125/250	125/500				
BW Dn [kHz]	125	500				
Potencia Tx Up [dBm]	+14dBm	+20dBm				
Potencia Tx Dn [dBm]	+14dBm	+27dBm				
SF Up	7-12	7-10				
Tasa de datos	250bps-50kbps	980bps-21.9kbps				

Tabla 4. Bandas frecuencias en LoRaWAN [25].

Varios son los aspectos descritos en los apartados anteriores que merecen una especial atención por su influencia directa en el modelo de consumo de energía del presente trabajo. La clase de dispositivo utilizado (A, B o C) tendrá un alto impacto en el consumo medio de los dispositivos debido al uso de diferentes mecanismos e intervalos de ventanas de recepción. El ancho de banda utilizado por los dispositivos repercutirá en el valor DR asociado a cada uno de los SF bajo el cual operan los dispositivos, así como su valor de sensibilidad. Por otro lado, cuanto mayor sea el valor SF de los dispositivos, mayor será su consumo en estado de transmisión debido al mayor tiempo de transmisión en el aire (TOA). El intervalo que define las ventanas de recepción, así como la limitación del *duty cycle* de acuerdo a diferentes bandas de operación tendrán también una influencia directa en el consumo debido a que determinan la lógica de los estados de operación de las capas inferiores del *stack* de los dispositivos. A su vez, el uso o no de mecanismos de control a través de comandos MAC permitirá o no el control del consumo de dispositivos finales de la red.

Por ello, será necesario tener presente cada uno de estos aspectos durante la simulación del modelo energético y localizar posibles limitaciones inherentes a la plataforma de simulación.

3.Entorno de simulación de red NS-3

3.1 Introducción

NS-3 [6] es un simulador de eventos discretos dirigido principalmente al campo de la investigación y al ámbito educativo, disponible públicamente bajo la licencia GNU GLPv2 para su uso y desarrollo. Con NS-3 se pretende desarrollar un entorno de simulación abierto que esté alineado con las necesidades de simulación de la investigación actual de redes, con un núcleo sólido, bien documentado, fácil de usar y depurar y que cubra las necesidades de todo el flujo de trabajo de simulación desde su configuración hasta la recopilación y análisis, así como alentar a la contribución de la comunidad para su integración continua, revisión y validación.

NS-3 posee una arquitectura modular que da suporte a una amplia variedad de redes, principalmente basadas en IP, tales como Wi-Fi, WiMAX, LTE, WAVE, 6LowPAN y una variedad de protocolos de enrutamiento estáticos o dinámicos como OLSR y AODV.

El entorno está constituido por un sistema de librerías de software que proporcionan un conjunto de modelos de simulación de red implementados como objetos en C++ y encapsulados en Python mediante los cuales el usuario interactúa implementando aplicaciones de más alto nivel en C++ o scripts en Python, en las cuales instancia los objetos que modelan la red deseada para así configurar el escenario de simulación de interés.

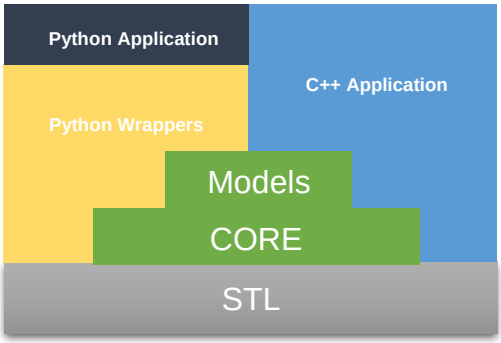


Figura 9. Modelo de programación de software en NS-3.

NS-3 se distribuye como código fuente, lo que implica que el sistema destino debe poseer un entorno de desarrollo de software para construir las librerías y posteriormente la aplicación de usuario de alto nivel. Pese a que los diferentes sub-sistemas de los que hacen uso los diferentes escenarios que constituyen la aplicación final de usuario podrían distribuirse como librerías pre-compiladas, vinculadas de forma estática o dinámica al programa principal, evitando la necesidad de compilar los diferentes módulos de software que integra el entorno.

La tendencia y la práctica actual es la de disponer del código fuente completo que posteriormente es modificado por el usuario y compilado para su posterior simulación y análisis en un entorno específico de desarrollo.

Desde el punto de vista del diseño de alto nivel, lo que distingue a NS-3 de otros simuladores de eventos discretos:

- **Énfasis en C++ y Python:** En contraste con otros simuladores que hacen uso de un lenguaje de modelado asociado a un dominio específico, NS-3 se aprovecha del uso extendido y soporte de lenguajes estándar como C++ y Python.
- ***Callback-driven events*:** Los eventos en la simulación son simples llamadas a funciones programadas para ejecutarse en un tiempo prescrito. Cualquier función puede convertirse en un evento y programarse mediante el uso de *callbacks*, a diferencia de la técnica que usa manejadores especializados de funciones que centralizan el procesamiento de eventos en cada objeto de simulación.
- **Núcleo flexible con capa asistente:** Un conjunto de APIs disponibles tanto en la capa de bajo nivel asociada al núcleo como en capas asistentes superiores ofrecen una alta flexibilidad y capacidad de configuración al usuario.
- **Reusabilidad de Software:** El entorno permite el uso de APIs de la familia de estándares POSIX para su ejecución directa en el espacio de usuario sin requerir cambios en el código de la aplicación gracias un *framework* propio, denominado *Direct Code Execution (*DCE). Los nodos, interfaces y objetos en NS-3 están diseñados de acuerdo con la arquitectura de red de sistemas Linux.
- **Gestión de configuración**: Un sistema integrado basado en atributos para la gestión de valores predeterminados e instanciados de los parámetros de la simulación, la integración con la línea de comandos y el uso de formatos y herramientas de documentación como XML, Doxygen y GTK, garantizan una eficaz gestión de configuración.
- **Independencia de IDE:** A diferencia de otros simuladores, el proyecto NS-3 no mantiene un IDE determinado para configurar, depurar, ejecutar y visualizar simulaciones en una única ventana, sino que está diseñado para trabajar con la línea de comandos e integrar herramientas de configuración y visualización según las necesidades del usuario.

3.2 Conceptualización y terminología

3.2.1 Estructura por capas

El código fuente de NS-3 sigue una estructura por capas modular donde únicamente existen dependencias entre módulos en sentido descendente.

El núcleo constituye la capa de más bajo nivel en la estructura, la cual alberga las clases fundamentales de la arquitectura de NS-3, como punteros inteligentes, gestión de tipado dinámico, atributos, *callbacks,* sistema de rastreo, registro de datos, etc. En la capa inmediatamente superior se encuentra la capa de red,

donde se realiza el modelado de los paquetes y objetos asociados (cabeceras, formatos, etiquetas, etc.), de los nodos que constituyen la topología de red, de los dispositivos de red, de direcciones asociadas a diferentes protocolos y estándares de red, así como otros elementos relacionados con el domino de red como colas, sockets, etc.

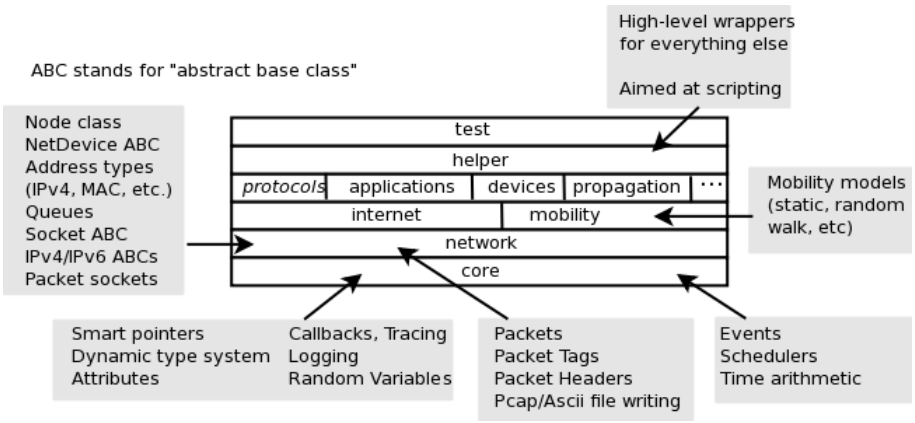


Figura 10.Estructura de software por capas en NS-3 [6].

Las capas de núcleo y de red constituyen a su vez el núcleo genérico de simulación bajo el cual se sustentan el resto de capas superiores, donde se lleva a cabo un proceso de abstracción de más alto nivel que modela diferentes aspectos que caracterizan a las diferentes redes, basadas o no en IP, y los diferentes escenarios objeto de simulación, tales como la capa de movilidad, la capa asociada al *stack* de internet, protocolos de enrutamiento, aplicaciones, tipos de dispositivos, modelos de propagación, etc., apoyados a su vez por las capas auxiliares del nivel superior que facilitan la configuración y manejo de la simulación, y las capas de test que permiten validar cada uno de los modelos de simulación que conforman NS-3.

3.2.2 Términos clave

A continuación, se detalla el conjunto de términos clave [6] en el modelado y simulación de redes, que caracterizan cada una de las capas que constituyen la arquitectura de software del entorno NS-3.

- **Nodo:** El nodo en NS-3 es la abstracción básica de cualquier elemento computacional sobre la topología de red. Pese a que en la jerga de Internet un dispositivo informático que forma parte de la topología de red suele denominarse *host* o dispositivo final, en NS-3 se utiliza el término nodo para proporcionalidad generalidad al concepto con independencia del tipo de red implementada en el simulador. La abstracción de este elemento está representada en C++ por la clase *Node*, la cual proporciona los métodos necesarios para gestionar las diferentes representaciones de los dispositivos que tienen lugar en la simulación, entendiéndolos como elementos computacionales aquellos a los cuales se les puede agregar

diferentes funcionalidades, traducido en modelos en el entorno NS-3, tales como tarjetas periféricas con controladores específicos, pilas de protocolos, aplicaciones, etc.

- **Canal:** Un canal en NS-3 es una ruta lógica sobre cual fluye la información entre los diferentes nodos de la topología de red. A cada uno de ellos se les agrega un modelo de canal determinado que es representado en C++ por la clase base *Channel* y clases derivadas, las cuales proporciona diferentes métodos para administras objetos de comunicación pertenecientes a la capa de subred de comunicación y agregarlos a los diferentes nodos de la simulación. Los modelos de canal pueden representar algo tan simple como un único hilo de comunicación, hasta modelos más complejos como conmutadores del estándar Ethernet, o un espacio tridimensional lleno de obstrucciones en el caso de redes inalámbricas. Ejemplos de canales en NS-3 son *CsmaChannel, PointToPointChannel, WifiChannel*, etc.

- **Dispositivo de red:** Un dispositivo de red en NS-3 hace referencia tanto al hardware que caracteriza una tarjeta de red determinada como al software de su controlador. El dispositivo de red es "instalado" en un nodo determinado para permitir que el nodo se comunique con otros nodos en el escenario de simulación a través del canal agregado a los mismos. La abstracción del dispositivo de red se representa en C++ con la clase base *NetDevice* y sus derivadas, las cuales proporcionan métodos para gestionar las conexiones en el entorno con nodos y canales de la red. Ejemplos de dispositivos de red en NS-3 son *CsmaNetDevice, PointToPointNetDevice, WifiNetDevice*, etc.

- **Auxiliares para la creación de topología (*topology helpers)*:** Los auxiliares en NS-3 son entidades que ayudan a gestionar la agregación e interconexión entre los diferentes objetos que tienen lugar en la simulación. Se encargan, por ejemplo, de la creación de dispositivos de red, de la agregación de direcciones físicas (MAC) a los mismos, de su instalación y configuración de la pila de protocolos en un nodo determinado, así como su conexión a un canal generado en el escenario de simulación. En topologías complejas donde deben conectarse múltiples dispositivos en canales multipunto o conectar redes de forma conjunta, existen además auxiliares de topología que proporcionan una serie de operaciones para la gestión de estas y facilitar la tarea al usuario.

- **Aplicación:** En el contexto de la ingeniería de sistemas el software puede dividirse en dos grandes grupos, el software del sistema o sistema operativo, y el software de aplicación. El primero gestiona los recursos informáticos, como memoria, ciclos de procesador, discos, red, etc., de acuerdo con algún modelo informático, constituyendo con ello la capa de abstracción entre la aplicación de usuario y las capas de bajo nivel de software o firmware de dispositivos y otros subsistemas. El segundo hace uso de los recursos anteriores a través de ciertas rutinas o APIs del sistema operativo para llevar a cabo funcionalidades específicas de alto nivel orientadas al usuario final. En NS-3 no existe un concepto como tal

de sistema operativo ni de los niveles de privilegios bajo los cuales se suele operar en el mismo para gestionar los recursos anteriores, sino que se centra en el concepto de software de aplicación, el cual puede ser agregado a los nodos que constituyen la topología de red para así crear diferentes escenarios en la simulación. La abstracción llevada a cabo por el software de aplicación es implementada en C++ a través de la clase base *Application* y clases derivadas. Ejemplo de aplicaciones en NS-3 son *ApplicationPacketProbe, BulkSendApplication, OnOffApplication, PacketLossCounter*, etc.

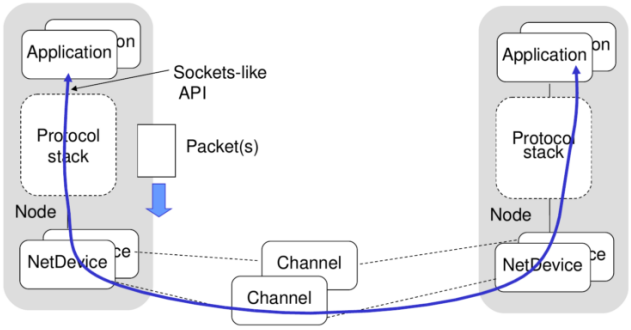


Figura 11. Ejemplo de un escenario básico en NS-3 [6].

3.3 Recursos y herramientas

El siguiente apartado tiene como objetivo proporcionar una breve descripción de las principales herramientas necesarias para trabajar sobre el entorno de simulación NS-3.

3.3.1 Waf

Waf [5] es una herramienta que permite configurar de forma automática la compilación e instalación de programas y librerías en entornos de desarrollo de software, bajo los lenguajes C++, OCami, D o Vala. Entre sus principales características cabe destacar:

- La ejecución de compiladores y scripts se lleva a cabo en procesos diferentes, y proporciona suporte para su configuración, así como de otras herramientas involucradas en el entorno.
- Posee memoria en la ejecución de procesos para aplicarlos únicamente ante cambios en los elementos de entrada, y permite su ejecución en paralelo de forma configurable según la máquina de compilación.
- Únicamente posee dependencias con Python, lenguaje en el cual está escrito, proporcionando además módulos reutilizables basados en él.
- Posee un esquema modular de configuración con análisis personalizable en línea de comandos y soporta la ejecución de pruebas en los programas tras el proceso de compilación.

3.3.2 Mercurial y Git

Para facilitar el desarrollo colaborativo, el mantenimiento, y la integración y continua de la plataforma, NS-3 soporta dos herramientas ampliamente utilizadas por la comunidad: Mercurial [3] y Git [2]. Ambas son sistemas de control de versiones multiplataforma, de software libre, con una arquitectura distribuida, a diferencia de otras herramientas como SVN, que hacen uso de SHA-1 para identificar revisiones, y de HTTP y SSH para el acceso a repositorios remotos.

3.3.3 GNUPlot

GNUPlot [35] es uno de los programas más populares de generación de gráficas utilizados por la comunidad. Está disponible bajo licencia de software libre, y soporta varios sistemas operativos, tales como Unix, Windows, Mac, Linux, etc. Con una interfaz de usuario basada en línea de comandos, permite la generación funciones y gráficas en dos o tres dimensiones que pueden ser representadas directamente en pantalla o almacenadas en archivos bajo diferentes formatos como *Portable Network Graphics* (PNG), *Encapsulated PostScript* (EPS), *Scalable Vector Graphics* (SVG), etc. Para realizar análisis en las simulaciones llevadas a cabo a través de NS-3, suele realizarse un uso intensivo de herramientas como GNUPlot, generando para ello archivos de salida durante las simulaciones con los datos de interés capturados, los cuales son usados por GNUPlot como elementos de entrada para generar representaciones gráficas a través de su lenguaje de intérprete. NS-3 dispone a su vez de un *framework* que facilita la integración entre la plataforma y GNUPlot.

3.4 Conclusión

La plataforma de simulación NS-3 (3.28) ha sido la utilizada para realizar la simulación del modelo energético del presente trabajo debido a varias de las características indicadas en los apartados anteriores. Su amplio uso en los campos de investigación actual de redes, así como su disponibilidad bajo licencia GNU GLPv2, facilita el acceso a la herramienta, así como al soporte de la comunidad ante cualquier incidencia. Dispone de una amplia variedad de librerías de software a utilizar en la implementación de los modelos de red promoviendo con ello diseños de alto nivel de cada uno de los elementos que forman parte de los módulos de red o en la simulación de escenarios. Su organización por capas y su estructura modular garantiza un alto grado de reusabilidad de software y permite derivar elementos asociados a un módulo de red a otro, realizando pequeñas adaptaciones. Su gestión de configuración basada en atributos de los elementos de red facilita la parametrización y monitorización de variables durante la simulación. La herramienta Waf, de la cual hace uso para llevar a cabo los procesos de configuración, compilación y ejecución de escenarios, permite un uso sencillo de la plataforma en los procesos de desarrollo, validación y despliegue. A su vez, el uso de Mercurial o Git como herramientas de control de versiones facilita el entorno colaborativo de la comunidad.

4. Simulación de LoRaWAN

En el presente capítulo se describen los aspectos más relevantes del modelo de simulación de una red LoRaWAN en NS-3 presentes en el repositorio [4] y descritos en las publicaciones relacionadas [1] y [43] en el cual se llevará a cabo la integración del modelo energético del presente trabajo.

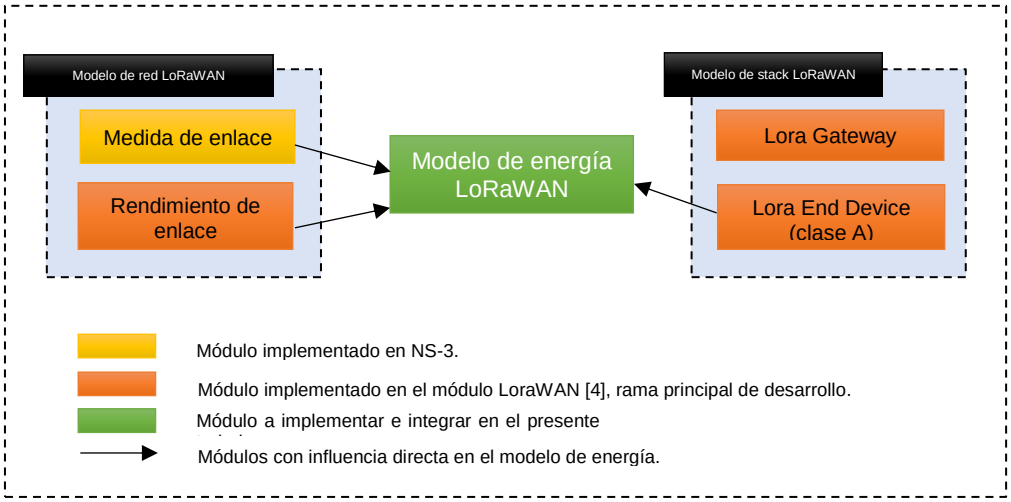


Figura 12. Componentes simulados en el módulo LoRaWAN de NS-3 [4]

Tal y como puede observarse en la figura 12, los diferentes componentes presentes en la simulación de la red conforman dos grandes bloques: el modelo de red LoRaWAN asociado a la cadena de transmisión entre elementos de una red LoRa, y el modelo de *stack* LoRaWAN asociado a cada una de las capas que constituyen la pila de protocolos presente en los dispositivos finales (ED) y puertas de enlace o *gateways* (GW).

Es importante señalar que tanto los componentes del modelo de red como los del modelo de *stack* LoRaWAN son elementos ya implementados en la rama principal del repositorio base [4], y que el presente trabajo se centra exclusivamente en la integración de un modelo energético en el *stack* para otorgarle capacidad de monitorización de consumo y eficiencia energética de los dispositivos finales de la red (ED). Por otro lado, cabe indicar que para la caracterización del modelo de medida de enlace en los escenarios simulados en el presente trabajo se hace uso de dos modelos genéricos implementados en NS-3 y no de los descritos en la publicación [43] asociada al repositorio [4] dada su no disponibilidad. Este hecho sin embargo no afecta al diseño del modelo energético, y tampoco tiene un impacto notable en los resultados de las simulaciones, ya que ambos modelos se rigen bajos los mismos principios.

4.1 Modelo de red LoRaWAN

El modelo de red LoRaWAN asume ciertas suposiciones y simplificaciones en la representación de una red de estas características para hacer viable computacionalmente su simulación bajo el entorno NS-3. Tal y como se detalla en [43] se basa principalmente en dos componentes que tienen como objetivo representar la cadena de transmisión en la red de una manera sencilla: el modelo

de medida de enlace usado para caracterizar los efectos de propagación en función de la potencia de la señal y promediar el desvanecimiento a pequeña a escala y efectos similares, y el modelo de rendimiento utilizado para determinar la probabilidad de recepción de un paquete de forma correcta asumiendo un modelo de sencilla complejidad a partir de la información de la intensidad del enlace, las fuentes de interferencia y otros efectos a nivel del sistema.

4.1.1 Modelo de medida de enlace

El modelo de medida de enlace tiene como principal objetivo, dada una transmisión entre un par transmisor-receptor, el estimar la intensidad de la señal que llega al receptor, teniendo en cuenta la potencia de transmisión, las ganancias de antena en el lado del transmisor y receptor, y la pérdida por propagación generada asociada al entorno de operación.

$$P_R[dBm] = P_T[dBm] + G_T[dB] + G_R[dB] - L[dB] \quad (4.1)$$

P_R: Potencia de recepción
P_T: Potencia de Transmisión
G_T: Ganancia antena transmisión
G_R: Ganancia antena recepción
L : Path loss variable en función de las condiciones de transmisión

Para caracterizar el término $L[dB]$ se hará uso del modelo de cálculo de pérdida por propagación para larga distancia sin obstáculos denominado *LogDistancePropagationLossModel* [44]así como el modelo híbrido de cálculo para entornos urbanos denominado *HybridyBuildingsPropagationLossModel* [40], en función de los escenarios de aplicación, ambos disponibles en NS-3.

El modelo de cálculo para larga distancia [44] será el utilizado en la simulación bajo escenarios de visión directa (LOS) y áreas extensas. El valor de las pérdidas por propagación viene dado por la siguiente expresión:

$$L[dB] = L_0 + 10nlog\left(\frac{d}{d_0}\right) \quad (4.2)$$

d_0: Distancia de referencia.
L_0: Path loss a la distancia de referencia.
d: Distancia de interés para el cálculo de path loss.

De acuerdo con lo establecido en [8], [9] y [43], y asumiendo una frecuencia de operación $f = 868MHz$, una elevación del receptor (GW) de $h = 15m$, una distancia de referencia de 1 m, las pérdidas de propagación en función de la distancia $d[Km]$ vienen dadas por:

$$L[dB] = 7.7 + 37.6log(d) \quad (4.3)$$

El modelo de cálculo híbrido disponible en NS-3 [44] será el utilizado en la simulación bajo escenarios de no visión directa (NLOS) en entornos urbanos donde se emplazará un *layout* de edificios emulando las condiciones que presenta un entorno de propagación de una gran ciudad. En él se combinan varios modelos diferentes para caracterizar las pérdidas por propagación en

función del entorno donde se produce la transmisión entre un emisor y un receptor concreto. El módulo contempla los modelos Okumura-Hata [39], ITU P. 1411 [36], ITU P.1238 [38], incluyendo pérdidas por penetración en edificios:

- El modelo ITU P.1238 modela las pérdidas por propagación en interiores en función del tipo de edificio (residencial, oficina, comercial, etc.) de acuerdo a la siguiente expresión:

$$L_{total} = 20\log(f) + Nlog(d) + L_f(n) - 28[dB] \qquad (4.4)$$

n: *Número de pisos* .
f: $frecuencia\ [Mhz]$.
d: $distancia\ [m]$.

$$N = \begin{cases} 28 & residencial \\ 30 & oficinas \\ 22 & comercial \end{cases} \qquad L_f = \begin{cases} 4n & residencial \\ 15+4(n-1) & oficinas \\ 6+3(n-1) & comercial \end{cases}$$

- En referencia al modelo ITU P.1411, éste considera la versión LOS para distancias superior a un umbral y NLOS para distancias superiores, en el cual se aplica el modelo de transmisión por encima de azoteas. En [42] pueden consultarse los valores específicos que considera el modelo.

- Adicionalmente, el módulo ofrece un conjunto adicional de elementos que caracterizan la propagación en entornos urbanos, otorgando con ello mayor realismo al modelo: tales como pérdidas por penetración a través de paredes exteriores e interiores (*EWL e IWL*), ganancia por altura, y desvanecimiento:
  - Pérdidas por penetración a través de paredes exteriores, *EWL,* en función del material utilizado*:*

$$L[dB] = \begin{cases} 4 & madera \\ 7 & hormigón\ con\ ventanas\ /\ 15\ hormigón\ sin\ ventanas \\ 12 & bloques\ de\ piedra \end{cases}$$

  - Pérdidas por penetración a través de paredes interiores, *IWL:*

$$L[dB] = L_{siw}(|x_1 - x_2| + |y_1 - y_2|) \qquad (4.5)$$

L_{siw}: $Pérdida\ asociada\ a\ pared\ interna, 5dB$.
x_1, x_2, y_1, y_2: $Número\ de\ habitaciones\ entre\ los\ ejes\ x\ e\ y, y\ usuarios\ 1\ y\ 2$

  - Ganancia en potencia de recepción debida a la altura del dispositivo, cuya expresión viene dada por:

$$GFH = n \cdot G_n \qquad (4.6)$$

G_n: $Ganancia\ debebido\ al\ incremento\ de\ altura\ por\ piso.\ G_n = 2\frac{dB}{m}$ [41].
n : $Número\ de\ pisos, valor\ distribuido\ uniformemente\ n = \{0,1,2,3,4\}$.

  - Pérdidas por desvanecimiento modeladas según una distribución log-normal con desviación estándar variable en función de la posición relativa (interior o exterior) del emisor y receptor. El modelo considera una media de desvanecimiento nulo y tres valores posibles de desviación estándar en fu
    - Asociada al exterior: 7 dB $X_o \sim (\mu_{o,} \sigma_O^2)$
    - Asociada al interior: 10 dB $X_I \sim (\mu_{I,} \sigma_I^2)$

- Asociada a la penetración en paredes externas: 5 dB $X_W \sim (\mu_W, \sigma_W^2)$

En el caso de transmisiones desde localizaciones externas a internas o viceversa, la desviación estándar viene dada por la contribución de la desviación asociada a nodos exteriores y a la penetración en paredes externas, dada la independencia entre ambas [40]:

$$X \sim N(\mu, \sigma^2), \qquad Y \sim N(v, \tau^2)$$
$$Z = X + Y \sim Z(\mu + v, \sigma^2 + \tau^2) \qquad (4.7)$$
$$\sigma_{IO} = \sqrt{\sigma_O^2 + \sigma_W^2}$$

X, Y, Z: *Variables aleatorias continuas, con distribuciones log-normales con medias* μ, v *y* $\mu + v$, *y varianzas* σ^2, τ^2 *y* $\sigma^2 + \tau^2$ *respectivamente.*
σ_{IO}: *Varianza asociada a transmisión entre exteriores e interiores o viceversa.*

En los escenarios urbanos simulados en el presente trabajo se establecerán bloques de tipo residencial de viviendas de 10 pisos, con paredes externas de hormigón y ventanas *EWL = 7dB*. Por otro lado, se configurará un umbral de 200m para la distinción entre la aplicación del modelo LOS y NLOS de ITU P.1411.

La lógica de aplicación del modelo de pérdidas híbrido se rige por el algoritmo representado en la figura 13. En primer lugar, se evalúa la posición del nodo transmisor y en función de ésta se evalúa la del nodo receptor y la distancia y localización relativa entre ambos para determinar el conjunto de contribuciones que caracterizan las pérdidas por propagación producidas en la transmisión.

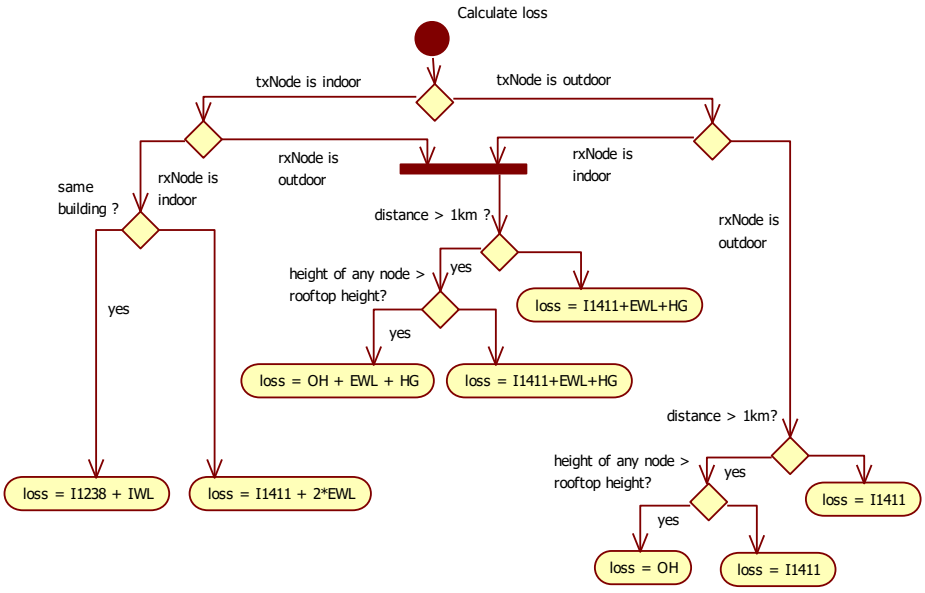


Figura 13. Algoritmo de cálculo pérdidas por propagación en el modelo híbrido de NS-3.

4.1.2 Modelo de rendimiento de enlace

El modelo de rendimiento de enlace implementado en [4] y descrito en [43] tiene como principal objetivo abstraer la implementación real de la cadena de trasmisión de la capa física (PHY) y facilitar los cálculos de interferencia en los escenarios planteados, tanto para los nodos de la red que operan como puertas de enlace (GW), como los que operan como dispositivos finales(ED). Para ello, se caracterizan aspectos como la sensibilidad de recepción e interferencia entre los nodos que operan en la red, haciendo uso de las hojas de datos de los dispositivos SX1301 (GW) y SX1272 (ED) de Semtech tomados como referencia para el modelo.

- **Sensibilidad de recepción:** Para caracterizar la sensibilidad de recepción de los nodos de la red, se distingue entre aquellos que operan como dispositivos finales (ED) y los que operan como puertas de enlace (GW). Asumiendo un ancho de banda de operación de $f = 125Khz$, los valores de sensibilidad de recepción de los GW, así como de los ED vienen discretizados de acuerdo a su valor de SF en la Tabla 5. En ella puede observarse como un incremento del SF implica una mayor sensibilidad y como para cada valor de SF los GW poseen una mayor sensibilidad en valor absoluto en comparación con los ED, cuantificado con un offset de +3dB. Para cada valor de la tabla ha de considerarse también el valor de la ganancia de antena de cada dispositivo, la cual mejora la sensibilidad de recepción.

Spreading Factor (SF)	Sensibilidad GW [dBm]	Sensibilidad ED [dBm]
7	-130.0	-127.0
8	-132.5	-129.5
9	-135.0	-132.0
10	-137.5	-134.5
11	-140.0	-137.0
12	-142.5	-139.5

Tabla 5. Relación entre SF y sensibilidad de los dispositivos LoRaWAN.

La condición que determina si un paquete enviado por un dispositivo que opera como emisor en un determinado momento es detectado por otro que opera como receptor, dado un valor de SF, es que la potencia de recepción sea superior a los umbrales de sensibilidad.

$$P_r[dBm] > S_i[dBm] \quad \forall\, SF_i \in [7,12] \qquad (4.8)$$

En caso de que la condición de detección se cumpla, el receptor cerrará la conexión y procesará la recepción del paquete, asumiendo que la potencia de recepción permanecerá constante durante todo el proceso.

- **Matriz SINR:** Para caracterizar la interferencia que tiene lugar entre los dispositivos, el módulo presente en [4] hace uso de la propiedad de ortogonalidad existente entre diferentes valores de SF para modelar la pérdida de paquetes debida a la interferencia de otras transmisiones LoRa. Dada la transmisión de un paquete con un $SF = SF_i$, afectada por

la interferencia de otra transmisión con un valor $SF = SF_j$ con $i \neq j$, se establece un margen $T_{i,j}$ del valor $SINR_{i,j}$ que debe satisfacer la primera para que el paquete puede ser decodificado.

$$SINR_{i,j} > T_{i,j} \qquad (4.9)$$

Contemplando un escenario con múltiples transmisiones interferentes que provienen de dispositivos finales y asumiendo un esquema de transmisión SISO, la expresión del valor $SINR_{i,j}$ viene dada por [43] y [14]:

$$SINR_{i,j} = \frac{P_{r,0}}{\sigma^2 + \sum_{l \epsilon I_j} P_{r,l}} \qquad (4.10)$$

$P_{r,0}$: *Potencia del paquete afectado por la interferencia asociada a un valor* $SF = SF_i$:
$P_{r,l}$: *Potencia del* $l-$*ésimo paquete interferente.*
I_j : *Conjunto de transmisión interferentes con un valor* $SF = SF_j$.

Y los márgenes $T_{i,j}$ vienen dados, de acuerdo a [43] y [13] por la siguiente matriz de valores límite:

$$T = \begin{array}{c|cccccc} - & \boldsymbol{SF_7} & \boldsymbol{SF_8} & \boldsymbol{SF_9} & \boldsymbol{SF_{10}} & \boldsymbol{SF_{11}} & \boldsymbol{SF_{12}} \\ \hline \boldsymbol{SF_7} & 6 & -16 & -18 & -19 & -19 & -20 \\ \boldsymbol{SF_8} & -24 & 6 & -20 & -22 & -22 & -22 \\ \boldsymbol{SF_9} & -27 & -27 & 6 & -23 & -25 & -25 \\ \boldsymbol{SF_{10}} & -30 & -30 & -30 & 6 & -26 & -28 \\ \boldsymbol{SF_{11}} & -33 & -33 & -33 & -33 & 6 & -29 \\ \boldsymbol{SF_{12}} & -36 & -36 & -36 & -36 & -36 & 6 \end{array} \qquad (4.11)$$

Mediante el uso de los valores de la matriz 4.11 se asume que las transmisiones se superponen completamente de forma sincronizada en el dominio temporal, sin embargo, no es lo que sucede en un escenario real, donde debido a técnicas de entrelazado utilizadas por las capas subyacentes de codificación de canal en la modulación LoRa, la interferencia entre dos señales x e y dadas está habitualmente solapada únicamente parcialmente. Por ello, el modelo de interferencia presente en [4] contempla además un ajuste adicional para considerar únicamente la potencia solapada temporalmente entre las señales en el cálculo de $SINR_{i,j}$:

$$P_{r,y_{interf}} = \frac{P_{r,y} \cdot t_{ol}}{T_x} \qquad (4.12)$$

$P_{r,y}$: *Potencia de la señal interferente.*
t_{ol} : *Intervalo temporal de solapamiento.*
Tx : *Intervalo temporal de la señal interferida.*

- **Modelo de recepción de un GW:** Para terminar de caracterizar el modelo de rendimiento de enlace presente en [4], se establecen una serie de suposiciones en los criterios de recepción de paquete de un dispositivo que opera como puerta de enlace en los escenarios de simulación:

- Cada dispositivo es capaz de emular 8 caminos de recepción diferentes en una misma antena.
- La frecuencia central de cada camino puede ser configurada individualmente.
- Cualquier SF puede ser recibido sin previa configuración en cualquiera de los canales de recepción.
- Si más de una ruta de recepción está agregada al mismo canal, se podrá gestionar en paralelo la recepción de tantos paquetes como cantidad de rutas de recepción de escucha, incluso si los paquetes son transmitidos utilizando un mismo SF.
- Si un paquete llega a un cierto canal y no hay caminos de recepción agregados al mismo, se pierde la recepción del paquete.

Tal y como puede observarse en los escenarios simulados en 6, el modelo de medida de enlace tiene influencia directa sobre la eficiencia energética de los ED, ya que las condiciones de propagación del entorno determinan el valor del SF bajo el cual operan los dispositivos, lo que a su vez establece un tiempo de transmisión en el aire (TOA) específico de las tramas transmitidas por los mismos y por ende unos intervalos de los modos de operación definidos y un consumo asociado a los mismos que variará en función del SF bajo el cual operan. El SF de operación es asignado a cada ED al inicio de la simulación estableciendo por cada ED el mínimo valor posible que garantice que la potencia de recepción del GW más cercano al mismo supere el valor de sensibilidad asociado al SF (tabla 5), teniendo en cuenta la localización del par ED-GW y el modelo de pérdidas por propagación utilizado en el escenario. De este modo se otorga el mínimo valor de SF que asegure la recepción de un paquete bajo condiciones de no interferencia, garantizando de esta manera condiciones óptimas de eficiencia energética en la red simulada en coherencia con la operatividad real de una red LoRaWAN. Por otro lado, la referencia al modelo de interferencia introduce las condiciones de recepción de paquetes consideradas en la implementación del módulo presente en [4], para su posible uso como medida de rendimiento de la red.

4.2 Modelo del *stack* LoRaWAN

El módulo de “LoRaWAN” presente en [4] simula la pila de protocolos de la especificación [24] tanto para dispositivos finales (ED) como gateways (GW). Para ello, siguiendo una estructura modular, una serie de clases caracterizan cada una de las capas que forman el *stack* de LoRaWAN, desde la capa *PHY* de más bajo nivel asociada a la modulación LoRa, hasta la capa de aplicación que gestiona la lógica de envío y generación de paquetes. Además, otras clases auxiliares permiten gestionar la configuración y la emulación de cada uno de los componentes del *stack*, así como otros parámetros asociados a la simulación de escenarios concretos. En la figura 14 puede observarse la interconexión vertical entre las clases que forman parte del *stack* de los dispositivos, así como la relación de las mismas con otras clases auxiliares para la gestión de parametrización y configuración de escenarios, y las asociadas al modelo de medida de enlace de la red descritas en 4.1.1.

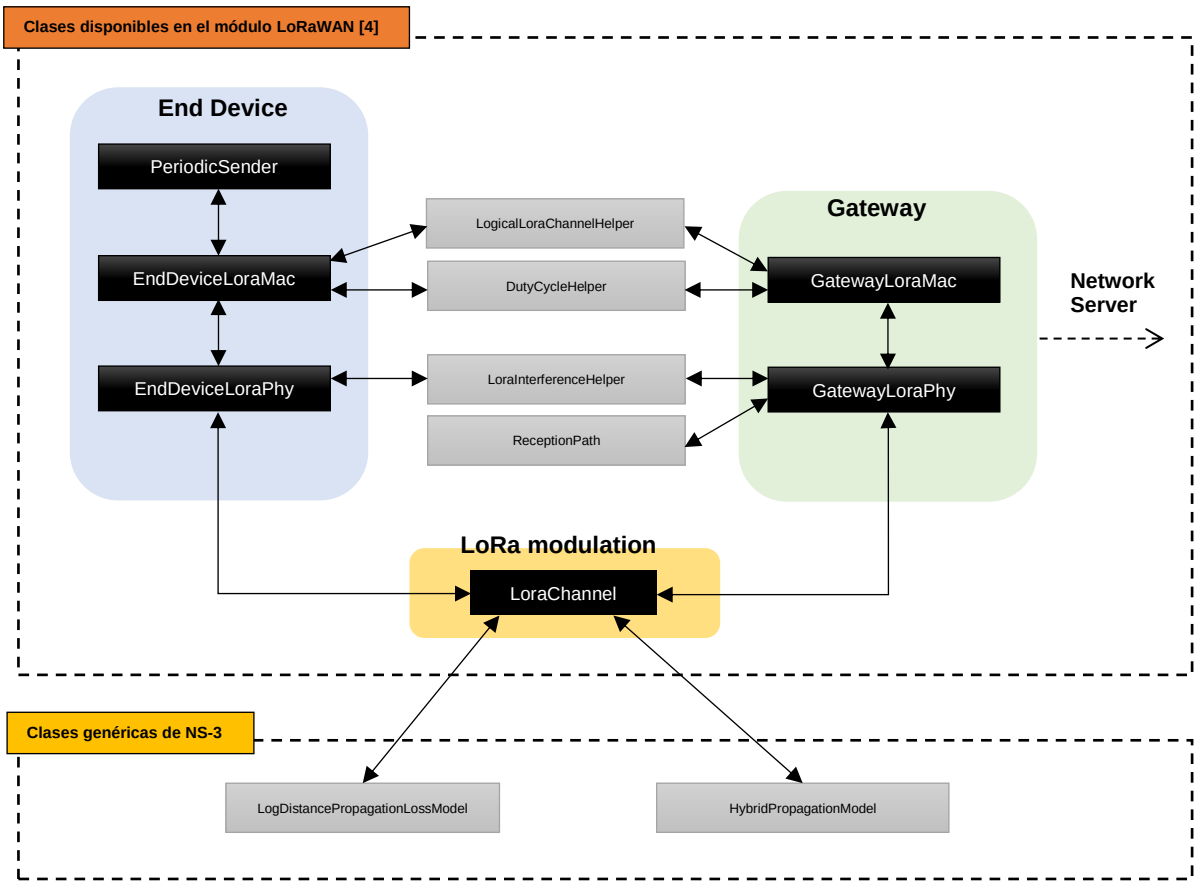


Figura 14. Arquitectura de alto nivel módulo LoRaWAN en NS-3.

A continuación, se describen las principales clases que conforman el módulo de LoRaWAN presente en [4], de las cuales se hará uso en los escenarios de simulación de 6.

4.2.1 LoraPhy

El modelo de la capa PHY, caracterizado por la clase *LoraPhy,* tiene en cuenta dos factores fundamentes en los esquemas de modulación LoRa, la sensibilidad y la ortogonalidad, para evaluar la validez de los procesos de transmisión, teniendo en cuenta el modo mediante el cual los dispositivos reales llevan a cabo la modulación.

La clase *LoraPhy* monitoriza el estado de los componentes de la capa física del *stack* a través del atributo *m_state,* el cual define cuatro modos de operación del dispositivo, de los cuales se hará uso de forma intensiva en el proceso de integración del modelo de energía en el módulo:

- **TX:** Asociado a la transmisión de un paquete.
- **RX:** Asociado a la recepción de un paquete.
- **STANDBY:** Asociado a la escucha activa de paquetes.
- **SLEEP:** Asociado al modo de bajo consumo donde no se lleva a cabo ningún proceso activo en el dispositivo.

A través de la clase auxiliar *LoraChannel*, la cual modela el canal de comunicación LoRa, se interconectan las diferentes capas físicas de todos los dispositivos involucrados en la comunicación de esta tecnología. Ésta dispone de una lista de clases *LoraPhy* agregadas a él, a las cuales notifica la existencia o no de conexiones entrantes, tal y como se establecen en otros modelos de simulación de red en el entorno NS-3. Las capas *LoraPhy* conectadas al canal, exponen un método público, *StartRecieve,* mediante el cual el canal inicia la recepción en una capa PHY de un dispositivo determinado. A su vez, se notifica a la clase auxiliar, *LoraInterferenceHelper*, la entrada de unos nuevos paquetes que operan bajo un SF determinado ya sean útiles o interferentes, para determinar si los primeros cumplen con las siguientes condiciones de recepción, y en su caso, iniciar el proceso de recepción del paquete:

- El estado de operación del receptor es STANDY en el momento en el que *StartReceive* es llamado.
- El receptor permanece en estado de escucha de acuerdo a la frecuencia y SF utilizados en la transmisión.
- La potencia de recepción supera el valor de sensibilidad establecido en la tabla 5 de acuerdo al SF utilizado en la transmisión.

El proceso de evaluación de paquetes entrantes puede observarse en la figura 15:

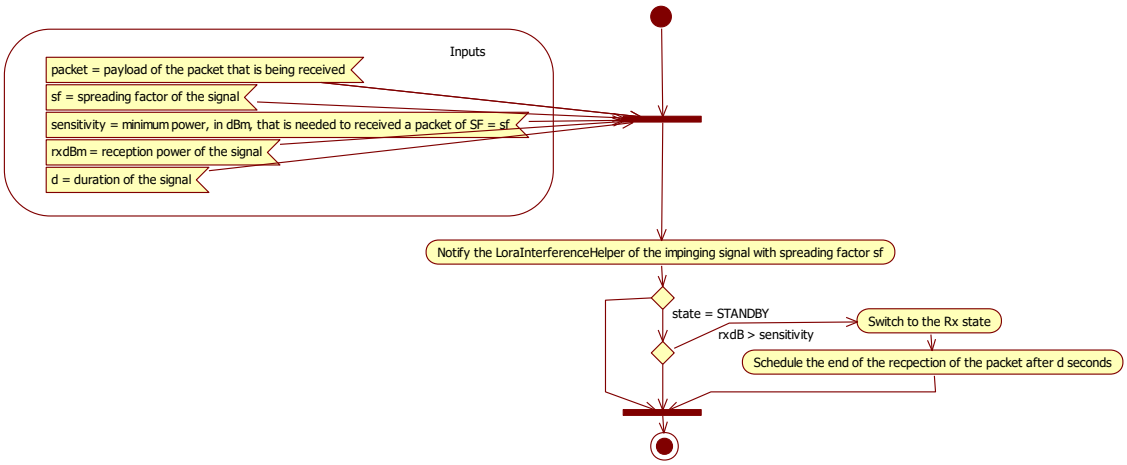


Figura 15. Inicio de recepción de un paquete en la capa PHY.

Si se cumplen las condiciones de recepción, la capa PHY programa un método, *EndReceive,* para su ejecución posterior tras la recepción del paquete, la cual hará uso del método *IsDestroyedByInterference* de la clase auxiliar *LoraInterferenceHelper* para determinar si el paquete se ha perdido o no debido a interferencia. El método *IsDestroyedByInterference* compara la potencia de recepción del paquete deseado con la energía de interferencia de los paquetes que se superponen considerando la contribución de cada uno de los SF, y posteriormente coteja el valor SNIR obtenido con la matriz de márgenes 4.11, en donde las filas identifican el SF de la señal útil y las columnas el SF de la señal interferente. En caso de que el SNIR cumpla el margen que se establece para

cada valor de SF, se lleva a cabo la recepción del paquete y se envía a la capa superior MAC, tal y como puede observarse en la figura 16:

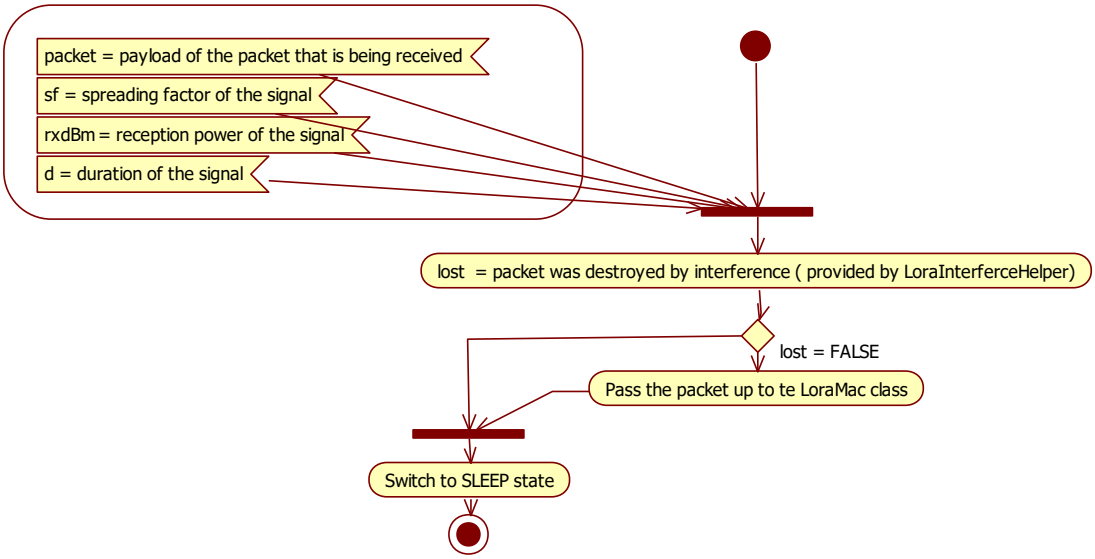


Figura 16. Fin de recepción de un paquete en la capa PHY.

La caracterización anterior representa el comportamiento de la capa PHY ante un sistema que posee una única ruta de escucha, como es el caso de los ED, caracterizados a través de la clase *EndDeviceLoraPhy*. Sin embargo, en los dispositivos que operan como GW, existen ocho rutas de recepción en paralelo. Estas rutas de recepción se modelan a través de la clase *ReceptionPath* que se comporta como un único *EndDeviceLoraPhy* para determinar si se establece de forma correcta la recepción de paquetes. La clase *GatewayLoraPhy* es la encargada de gestionar cada una de las rutas de recepción instanciadas, la cual, ante la llegada de un paquete, elige un camino de recepción libre, si lo hay, para procesar su recepción, y consultar a la clase auxiliar *LoraInterferenceHelper* si es posible llevar a cabo la recepción del paquete, siguiendo el proceso análogo al de una única ruta.

4.2.2 LoraMac

El modelo de capa MAC caracteriza los aspectos más relevantes de la especificación LoRaWAN [24]. Para ello, hace uso de una serie de clases que se encargan del manejo de encabezados, canales lógicos, cálculos de ciclos de trabajo y gestión del uso de la banda frecuencial.

La clase auxiliar *LogicalLoraChannelHelper* lleva a cabo la monitorización de cada uno de los canales lógicos disponibles y de las sub-bandas asociadas, haciendo uso de clases subyacentes como *LogicalLoraChannel* y *SubBand.* A su vez, a través de la clase auxiliar *DutyCycleHelper* se registran todas las transmisiones que son llevadas a cabo en cada canal lógico para determinar cuándo es posible realizar una nueva transmisión y cumplir así con las

limitaciones de ciclo de trabajo de transmisión establecidas en la especificación LoRaWAN de acuerdo con la regulación vigente.

Dada una transmisión de duración t_{air} llevada a cabo por un dispositivo en un canal Lora, el tiempo t_{off} que el dispositivo debe permanecer en estado de reposo hasta la siguiente transmisión, viene dado por la siguiente expresión:

$$DC = \frac{t_{air}}{t_{air} + t_{off}} \rightarrow \quad t_{off} = \frac{t_{air}}{DC} - t_{air} \qquad (4.13)$$

DC: $Duty\ Cycle$.

La secuencia de envío completa puede observarse en la figura 17:

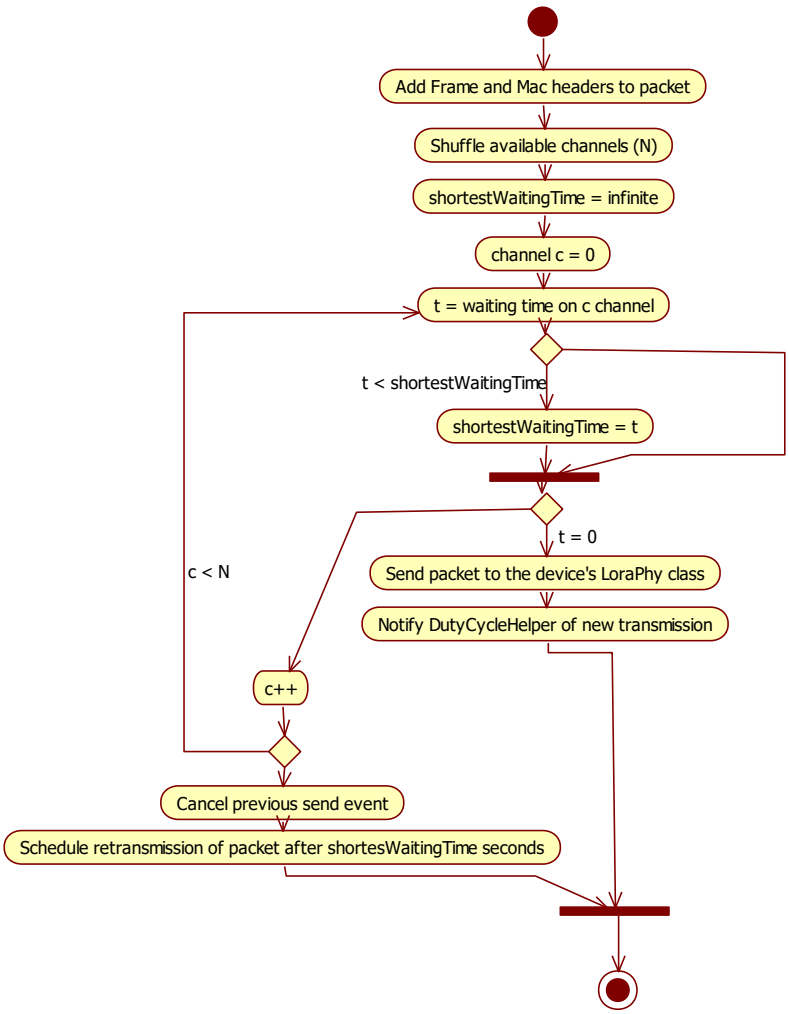


Figura 17. Procedimiento de envío en *LoraMac.*

La estructura de tramas definida por la especificación LoRaWAN es implementada a través de las clases *LoraMacHeader* y *LoraFrameHeader*. Ésta última hace uso, a su vez, de las clases subyacentes *MacCommand* y *LoraDeviceAddress* para llevar a cabo procesos de serialización, deserialización, interpretación de comandos MAC, y direccionamiento.

4.2.3 PeriodicSender

La lógica de generación y envío de paquetes hacia el servidor de la red LoRaWAN por parte de los dispositivos finales (ED), se lleva a cabo a través de la clase *PeriodicSender*, derivada de la clase base *Application* de NS-3, que caracteriza las capas superiores del *stack*. A través de ella, se generan paquetes con un contenido binario de ceros en su totalidad, y con un *payload* de tamaño configurable, y se transfieren a la capa inmediatamente inferior del *stack,* la capa MAC.

Una vez instanciada la aplicación en el escenario, los diferentes ED pueden subscribirse a la misma para emular el envío de datos hacia un servidor de red para llevar a cabo el análisis de los mismos tal y como se llevaría a cabo en un escenario real bajo una infraestructura de red LoRaWAN.

El periodo de transmisión de paquetes a la capa inferior puede ser configurado de acuerdo a un valor fijo o establecerse un valor aleatorio haciendo uso de una distribución uniforme $d \sim U[0; period]$ para asignar valores de ciclos de transmisión diferentes a cada uno de los ED que están subscritos a la aplicación en el escenario donde ésta es instanciada.

4.2.4 Otras clases

Además de las clases anteriores que modelan las diferentes capas del *stack* de comunicación de LoRaWAN, la simulación hace uso de una serie de clases auxiliares que son necesarias para llevar a cabo la simulación de red en el entorno NS-3, de acuerdo a la estructura por capas descrita en 3.2.1. Entre ellas, cabe destacar la clase *LoraChannel*, responsable de modelar el canal inalámbrico de la red, capturando y entregando paquetes asociados a una potencia de transmisión determinada a la capa *LoraPhy*, la clase *LoraNetDevice*, que modela “una tarjeta de red Lora”, asociando a un nodo determinado todos los dispositivos propios del *stack* pertenecientes a la capas *LoraPhy* y *LoraMac*, y las clases auxiliares *LoraHelper, LoraPhyHelper, LoraMacHelper,* que llevan a cabo la configuración y parametrización de los elementos del *stack*. Por otro lado, la clase *OneShotSender* puede ser utilizada en escenarios con topologías sencillas o con propósitos de validación para generar envíos de un único paquete bajo un determinado *timestamp* en la simulación.

5.Integración de un modelo de energía

Analizado el modelo de red y la arquitectura LoRaWAN implementados [4] , de los cuales hace uso el presente trabajo como modelo base de simulación de red, los siguientes apartados tienen como objetivo describir el modelo teórico, el diseño de software y la codificación del modelo energético a integrar en el mismo. Tras el proceso de codificación se detalla a su vez el proceso de validación implementado y se definen los escenarios de aplicación en los cuales se llevará a cabo un posterior análisis de acuerdo a unas métricas definidas.

5.1 Alcance

De acuerdo con los objetivos planteados en el presente trabajo y considerando las limitaciones temporales en su desarrollo e implementación, los siguientes puntos definen el alcance contemplado en el proceso de integración del modelo energía:

- Definir el modelo teórico de fuente de energía que caracterice el suministro y almacenamiento energético de los dispositivos finales (ED) de una red LoRaWAN.
- Definir el modelo teórico de consumo energético de los transceptores de los ED, identificándolos como principal elemento de consumo en el sistema, en función de diferentes modos de operación del dispositivo.
- Establecer un conjunto de métricas que permitan validar el modelo de consumo y almacenamiento energético implementado.
- Definir diferentes escenarios de aplicación del modelo que permitan analizar la eficiencia y autonomía de los dispositivos en entornos reales.
- Integrar herramientas de recolección de datos en el modelo que permitan realizar su análisis posterior tras la simulación.
- Analizar los datos obtenidos en la simulación y proponer vías de optimización de eficiencia energética de los dispositivos.

En cuanto a las limitaciones del modelo, señalar los siguientes puntos:

- El modelo de fuente de energía asume ciertas aproximaciones para reducir la carga computacional y no afectar al rendimiento de la simulación.
- El modelo de consumo de corriente contempla márgenes de errores asociados a la diversidad de dispositivos en el mercado, así como a la confidencialidad del modelo, lo que dificulta su caracterización de acuerdo a un modelo teórico exacto.
- Tanto el modelo de fuente de energía como el de consumo se aplica únicamente a dispositivos finales (ED) de la red LoRaWAN, dejando al margen a los dispositivos *gateways* (GW).
- El modelo implementado en [4] soporta únicamente operación de dispositivos finales de clase A, asociada a mayores restricciones en consumo, lo que deja al margen el estudio de la eficiencia energética de los dispositivos de clase B y C.

- Los escenarios contemplados en el presente trabajo consideran únicamente tráfico en sentido ascendente (UP) correspondiente al envío de datos de los dispositivos ED a los GW. Esta limitación viene determinada por la no disponibilidad en [4] de la infraestructura necesaria para realizar mecanismos de control de red, tales como ADR, monitorización del margen de enlace, control del *Duty Cycle,* control de las ventanas de recepción, etc. Este hecho, sin embargo, no supone una gran limitación en el análisis del consumo de los dispositivos dado que la mayoría del tráfico en una red LoRaWAN es generado en sentido ascendente, aunque bien es cierto que, la simulación del sentido descendente de enlace permitiría modificar ciertos parámetros en los dispositivos para mejorar su eficiencia energética. Una vez que su implementación se encuentre disponible en [4], el acoplo del modelo de energía del presente trabajo se realiza de forma automática, dado que las interfaces con el resto de componentes del *stack* siguen el modelo de energía validado en la plataforma NS-3.

5.2 Trabajos relacionados

Debido a la propiedad de la tecnología LoRa por parte de Semtech, no resulta sencillo encontrar información acerca del modelo teórico del consumo de energía del hardware utilizado en los dispositivos finales (ED). El presente trabajo toma como referencia estudios y análisis recientes, así como la información presente en las hojas de datos de los dispositivos para establecer un modelo teórico del consumo de los dispositivos para su simulación en un escenario lo más cercano posible a la realidad. En los diferentes estudios, así como en el *benchmark* proporcionado por los fabricantes en las hojas de datos, se establecen principalmente cuatro modos de funcionamiento del dispositivo: TX, RX, STANDBY y SLEEP, asociados respectivamente a los modos de transmisión, recepción, reposo y suspensión, y modelados en la implementación de referencia donde aplicar el modelo energético, tal y como se describe en 4.2.1.

En [26] se lleva a cabo una comparación del consumo de los dispositivos iM880B-L [27], basados en el microcontrolador Cortex-M3 y el transceptor LoRa SX1272 de Semtech en un escenario real operando en la banda de 868 MHz con un ancho de banda de 125Khz. El estudio compara diferentes parámetros de consumo en las diferentes clases bajo las cuales puede operar un dispositivo ED en LoRaWAN. El consumo de corriente del dispositivo en los modos TX, RX, STANDBY son comparables a lo especificado por las hojas de datos, es decir 118mA, 11mA y 1.6mA. Sin embargo, el valor obtenido en el modo SLEEP es de 1.27 mA, tres órdenes de magnitud superior a lo esperado (1.8μA), pero este hecho se asocia a una errónea configuración del dispositivo en operación de bajo consumo, configurado en modo STANDBY y no en SLEEP, como cabría esperar. A su vez se hace una estimación del tiempo de vida de batería, considerando los valores teóricos de consumo para unos valores de SF de 7 y 11 y parametrizando el tamaño del paquete utilizado en la transmisión. En las figuras 18 y 19 puede observarse como la carga necesaria para una operación de 10 años puede llegar a 4000 mAh con un valor de SF=7, y hasta 12000 con un valor SF=11.

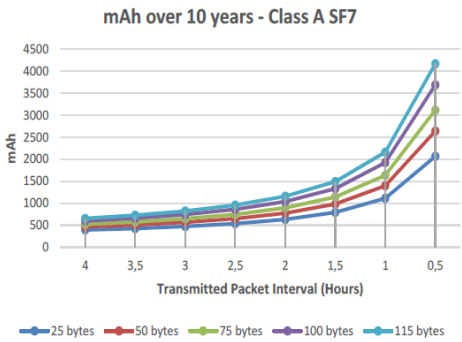


Figura 18. Evolución del consumo en dispositivos clase A y SF=7 [26].

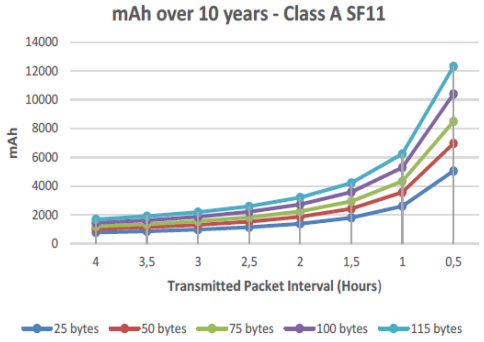


Figura 19. Evolución del consumo en dispositivos clase A y SF=11 [26].

En [28] se realiza un análisis en detalle del consumo de energía de dispositivos ED de clase A considerando el impacto en el mismo la variación de diferentes parámetros tales como el DR, el *payload,* el uso de ACK, el BER. Entre las conclusiones derivadas del estudio, se establece que un dispositivo ED operando con una batería de 2400 mAh que envía tramas con un ciclo de 5 minutos, logra un ciclo de vida de 1 año. A medida que aumenta el periodo de notificación, la duración teórica del ED tiende asintóticamente a aproximadamente 6 años bajo las condiciones consideradas. Se establece también que, dado el consumo típico de 1μA en estado de suspensión de los ED, los dispositivos pueden operar de forma autónoma durante años a través del uso de “pilas de botón”, las cuales contienen una capacidad un orden inferior de las consideradas en el estudio.

En [29], se evalúan las características de consumo energético en diferentes escenarios en los dispositivos SX1278 de Semtech. En el estudio se observa como el valor de corriente del dispositivo DRF1278F del fabricante Dorki [30] operando en modo TX, un ancho de banda de 125KHz oscila entre 54 mA para un SF=9, y 95 mA para un SF=12.

Con la simulación del consumo energético implementada en el presente trabajo se pretende obtener resultados coherentes con los estudios anteriores, así como con los *benchmark* de consumo disponible en las hojas de datos de los dispositivos, teniendo presente las suposiciones y aproximaciones contemplados en el modelo para permitir que la simulación sea factible en el entorno planteado.

5.3 Análisis y diseño de la solución

Para llevar a cabo la integración del modelo de energía, se hará uso del *framework* o módulo de energía presente en NS-3 [45] basado en un modelo dual de caracterización de fuentes de energía y de consumo de corriente de dispositivos. El módulo dispone de varias clases de referencia con diferentes mecanismos de abstracción que serán utilizadas como clases base para implementar la caracterización específica de fuentes de energía y consumo de los dispositivos ED que operan en una red LoRaWAN, a través de mecanismos de herencia y que proporcionará escalabilidad al software y facilitará la

incorporación de futuras implementaciones. Además, se utilizará como implementación de referencia el modelo energético llevado a cabo en el módulo Wifi de NS-3 [46], basado en el consumo del transceptor del dispositivo, el cual se adaptará al modelo de LoRa, estrategia análoga a la seguida en el modelo de energía iniciado en la rama de desarrollo de [4] basado en un modelo de energía básico de consumo constante.

5.3.1 *Framework* de energía en NS-3

El módulo de energía presente NS-3 dispone de dos clases fundamentales *EnergySource* y *DeviceEnergyModel* que están interconectadas a través de una serie de interfaces predefinidas. *EnergySource* representa diferentes tipos de fuentes de energía, como las baterías de Ion-litio, alcalinas, ácido-plomo reguladas por válvula (VRLA), de níquel, etc., que son almacenadas en una clase contendor *EnergySourceContainer*. *DeviceEnergyModel* representa el modelo de consumo de energía de los diferentes dispositivos agregados a un nodo en la simulación, almacenados de forma análoga al modelo de fuente en una clase contenedor *DeviceEnergyModelContainer*. El proceso de suministro de energía está separado del proceso de consumo para que puedan modelarse individualmente. A través de las interfaces implementadas en el módulo, los modelos de consumo notifican el gasto energético a los modelos de fuente de energía y éstos notifican a los primeros cualquier evento que se produce en la fuente del dispositivo, como por ejemplo el consumo completo de la fuente de energía, un reemplazo de la batería, etc.

Entre el conjunto de APIs proporcionadas por la clase abstracta *EnergySource* cabe destacar las siguientes:

- *GetInitialEnergy*: Suministra el valor inicial de energía almacenado en la fuente.
- *GetRemainingEnergy*: Facilita el valor remanente de energía en la fuente.
- *GetEnergyFraction*: Proporciona la relación (fracción) entre el valor de energía actual y el valor inicial.
- *UpdateEnergySource*: Actualiza el valor restante de energía en la fuente. Es el método usado por la clase *DeviceEnergyModel* como principal interfaz para actualizar el estado de la fuente de energía.
- *AppendDeviceEnergyModel:* Añade un nuevo modelo de consumo de energía a la fuente, para caracterizar su suministro energético.
- *CalculateTotalCurrent:* Calcula el actual consumo de corriente demandado en la fuente teniendo en cuenta todos los dispositivos agregados a ella.
- *NotifyEnergyDrained:* Envía la notificación de “energía agotada” a los dispositivos agregados a la fuente.

Entre el conjunto de APIs proporcionadas por la clase abstracta *DeviceEnergy Model* cabe destacar las siguientes:

- *SetEnergySource*: Asocia el modelo de energía del dispositivo con un modelo de energía de fuente determinado.

- *ChangeState*: Notifica al modelo de cambios en los estados de operación del dispositivo.
- *GetCurrentA*: Proporciona el consumo de corriente del dispositivo en su estado de operación actual.
- *HandleEnergyDepletion*: Función llamada por la fuente de energía a la cual está agregada el modelo de energía del dispositivo, una vez que la fuente está agotada.
- *GetTotalEnergyConsumption*: Proporciona el consumo total del dispositivo en momentos discretos de la simulación.

En la figura 20 puede observarse los principales métodos de las clases base principales y la relación entre ellas.

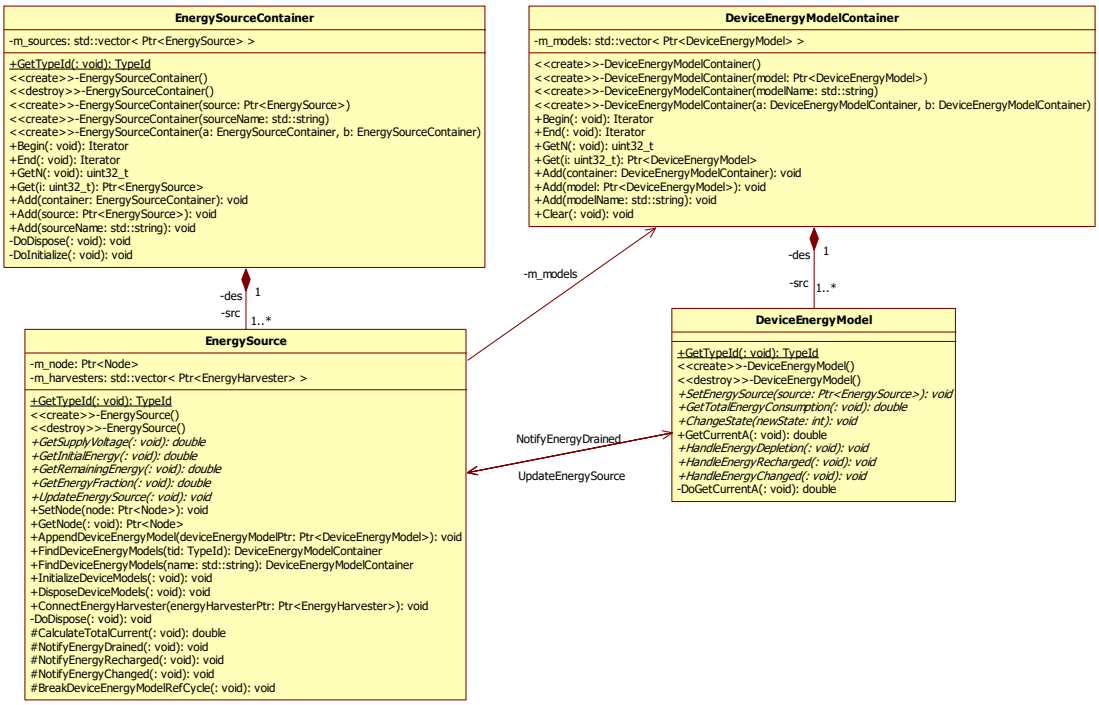


Figura 20. Diagrama de clases en el *framework* de Energía en NS-3.

5.3.2 Modelo de energía de LoRaWAN

La solución adoptada para integrar el modelo de energía hará uso del *framework* de energía de NS-3 fundamentado en un modelo dual fuente-consumo a partir del cual se particularizarán los parámetros de energía de los dispositivos en una red LoRaWAN mediante la creación de dos clases principales heredadas *LoraEnergySource* y *LoraRadioEnergyModel*, así como de otras clases auxiliares que facilitarán la configuración y la monitorización de los parámetros de energía durante la simulación. En primer lugar, se detalla el modelo teórico de fuente de energía y consumo de dispositivo y posteriormente se describe el diseño y arquitectura de software que caracterizará el modelo energético durante la simulación.

5.3.2.1 Modelo de fuente de energía

Varios son los sistemas de batería utilizados en el contexto IoT para proporcionar energía a los dispositivos de la red que funcionan bajo sistemas de alimentación autónoma. Algunos de ellos son los sistemas que utilizan químicos alcalinos, los basados en ion-litio, los basados en litio-metal, las basados en níquel o los sistemas de batería VRLA. El modelo utilizado en el presente trabajo está basado en una idealización del modelo de batería ion-litio, dada su popularidad en el mercado [31], [32], [33] debido a sus prestaciones y tamaño.

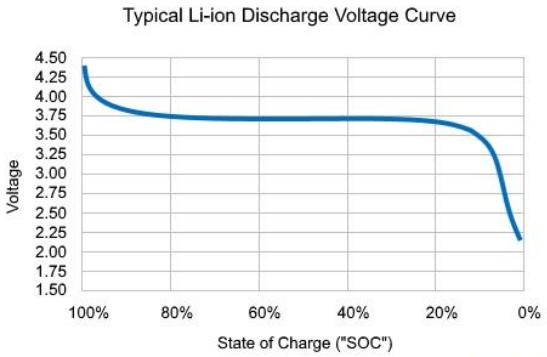


Figura 21. Perfil de descarga de una batería Ion-Litio. Voltaje – *SoC* [47]

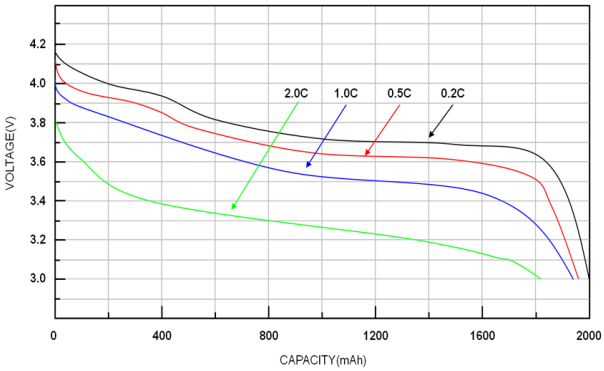


Figura 22. Perfil de descarga de una batería Ion-Litio. Voltaje – *Capacidad* [34].

El valor de tensión proporcionado por la batería en el modelo de fuente de energía será de $V = 3.7\,V$, dado su uso común y compatibilidad con la mayoría de los rangos de tensión de alimentación de la mayoría de los dispositivos, mientras que la capacidad será un parámetro configurable en los escenarios de simulación. El perfil de descarga de los sistemas basados en ion-litio, tal y como puede observarse en la figuras 21 y 22, se caracteriza por disponer de tres zonas con una evolución de descarga diferente. La primera está asociada a una descarga inicial exponencial, la segunda sigue una evolución de descarga cuasi lineal correspondiente al área nominal de la curva, y la última sigue una descarga no lineal correspondiente al último intervalo de operación de la batería.

Sin embargo, el modelo de fuente de energía en la simulación se basará en un modelo aproximado que contendrá un perfil de descarga lineal con un nivel de tensión constante durante todo el intervalo de operación de la fuente de energía (figuras 23 y 24).

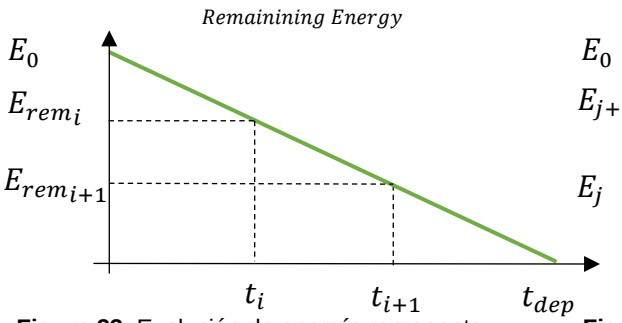


Figura 23. Evolución de energía remanente.

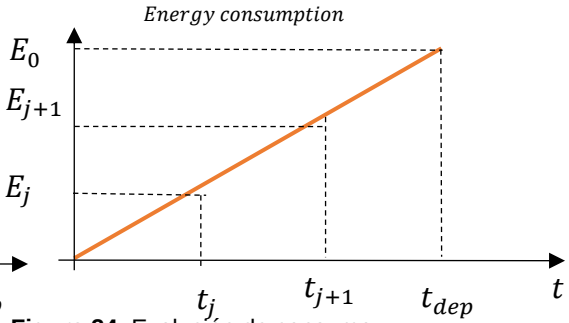


Figura 24. Evolución de consumo.

De este modo se reduce la carga computacional en la simulación y la complejidad de la codificación del modelo, teniendo en cuenta que se producen desviaciones con respecto al modelo real únicamente en las zonas de descarga inicial y final.

El modelo lineal presentará un valor inicial de energía E_0 configurable en el escenario de simulación que irá reduciéndose ante notificaciones de consumo de los dispositivos agregados a la fuente de energía. Considerando un valor de consumo de energía E_i absoluto en una marca temporal de simulación t_i, correspondiente a un consumo instantáneo de corriente I_i, el consumo de energía absoluto en la siguiente marca temporal t_{i+1} del proceso de actualización de la fuente, así como la energía remanente, vendrán dados por:

$$E_{i+1} = E_i + P_i \cdot (t_{i+1} - t_i) \qquad (5.1)$$
$$E_{i+1} = E_i + V \cdot I_i(t_{i+1} - t_i) \qquad (5.2)$$
$$E_{rem_{i+1}} = E_0 - E_{i+1} \qquad (5.3)$$

La clase que representará este modelo matemático será *LoraEnergySource*, heredada de la clase base *EnergySource* y basada en la clase *EnergySourceBasic* disponible en el *framework* de Energía de NS-3, a la cual se han añadido una serie de interfaces para configurar y monitorizar el estado de la batería en unidades de mAh.

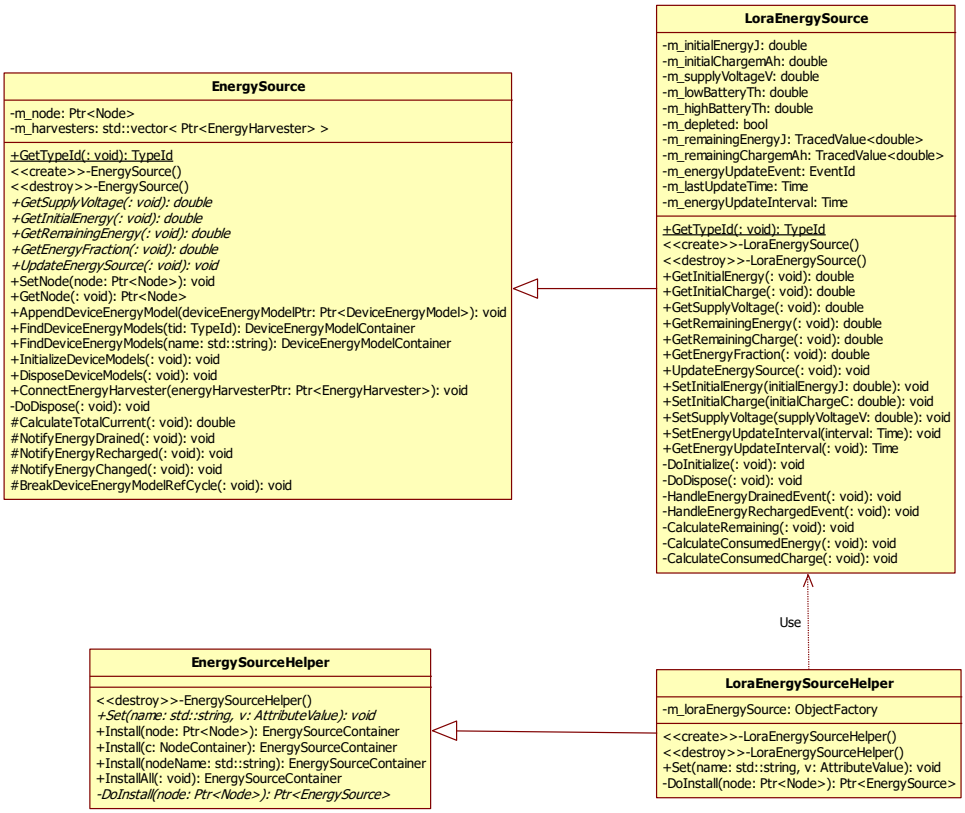


Figura 25. Diagrama de clases del modelo de fuente de energía de LoraWAN.

A su vez, se hará uso de la clase auxiliar *LoraEnergySourceHelper*, heredada de la clase base *EnergySourceHelper,* para la creación y configuración de las fuentes de energía instanciadas en los escenarios de simulación. En la figura 25, puede observarse las interfaces públicas de cada una de las clases, así como la relación entre ellas.

5.3.2.2 Modelo de consumo de dispositivo

Caracterizado el modelo de fuente de energía a utilizar en el entorno de simulación, será necesario establecer ahora un modelo de consumo de los dispositivos agregados a ella para proporcionarle los valores de corriente correspondientes a los diferentes modos de operación: TX, RX, STANDBY y SLEEP. En el cómputo de consumo de corriente se ha considerado únicamente el transceptor RF, y un consumo constante de la unidad MCU encargada del procesado, despreciando con ello la variabilidad en el consumo de la misma dependiendo de los módulos operativos que disponga (ADC, SPI, I2C, UART, etc.).

En referencia al transceptor RF, dado que el módulo “LoRaWAN” [4] está basado en el dispositivo ED Semtech SX1272, que cuenta con gran popularidad en el mercado, se utilizará esta variante como modelo de referencia en la simulación. En la tabla 6 pueden verse los valores de consumo de corriente correspondientes a cada modo de operación del dispositivo extraídos de la hoja de datos del fabricante [7].

Modo	Condiciones	Consumo típico
STANDBY	N/A	1.4 mA
TX	+20 dBm +17 dBm +13 dBm +7 dBm	125 mA 90 mA 28 mA 18 mA
RX	*LnaBoost On*	11.2 mA
SLEEP	N/A	1.8 μA

Tabla 6. Consumo de los dispositivos ED en LoRaWAN.

Los modos de operación de STANDBY, RX y SLEEP poseen un único valor de consumo de corriente con independencia de otros factores, sin embargo, el valor de consumo en el modo TX posee una dependencia directa con el valor de la potencia de transmisión utilizada en el envío de paquetes. Para establecer el consumo de corriente en el modo de operación TX en función del valor de potencia de transmisión, se realizará una interpolación lineal a partir de los valores disponibles en la hoja de datos:

$$I_{TX}(P[dBm]|P_1;P_2) = I(P_1) + \frac{I(P_2) - I(P_1)}{P_2 - P_1}(P - P_1) \qquad (5.4)$$

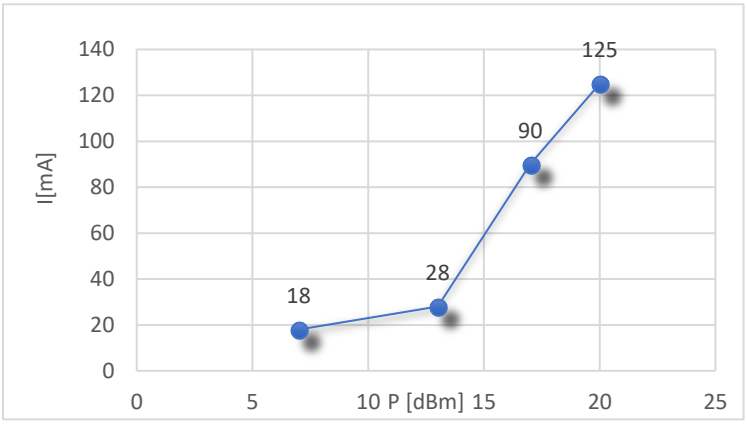


Figura 26. Interpolación del valor de consumo en modo TX de los dispositivos ED.

Caracterizados los valores del consumo de corriente, $I_{STANDBY} = 1.4\,\text{mA}$, $I_{SLEEP} = 0.1\,\mu A$, $I_{RX} = 11.2\,\text{mA}$ e $I_{TX}(P[dBm]|P_1;P_2)$, deberá definirse un modelo de software para llevar a cabo la monitorización del consumo de corriente de los transceptores de los dispositivos asociado a un modo de operación determinado para proporcionar los datos a la clase *LoraEnergySource* y llevar a cabo la actualización del estado de la batería.

La clase *LoraRadioEnergyModel*, heredada de la clase base *DeviceEnergyModel*, será la encargada del aprovisionamiento de datos de consumo de los dispositivos a la clase *LoraEnergySource* apoyándose en una serie de clases auxiliares que permitirán el acoplamiento entre el modelo energético y el modelo del *stack* de comunicación de LoRaWAN implementado en [4]. Entre los métodos implementados en la misma, cabe destacar:

- *Get<Mode>EnergyConsumption:* Proporciona el consumo asociado a un modo de operación durante todo el tiempo de simulación.
- *GetTotalEnergyConsumption:* Provee el consumo total asociado al conjunto de modos de operación durante todo el tiempo de simulación.
- *GetTotal<Mode>Time:* Contabiliza el tiempo de funcionamiento del dispositivo en cada modo de operación.
- *GetT<Mode>CurrentA:* Proporciona el valora de consumo de corriente asociado a un modo de operación.
- *CalcTxCurrentFromModel:* Calcula el consumo de corriente en modo TX de acuerdo al modelo de consumo utilizado (modelo interpolado).

Para realizar el cálculo y proveer el consumo de corriente de los dispositivos en modo TX, se utilizará una clase genérica, *LoraConsumptionModel,* que dispondrá de las interfaces para la provisión de los datos de consumo en función de la potencia de operación y la clase derivada, *InterpolatedLoraConsumptionModel*, caracterizará el modelo de consumo de corriente basado en interpolación.

La monitorización del modo de operación de los dispositivos será llevada a cabo por la clase *LoraPhyListener*, y su derivada, *LoraRadioEnergyModePhyListener*. La primera modela la interfaz a incorporar en el modelo *LoraPhy* del *stack*

LoRaWAN para notificar las transiciones de los modos de operación de los dispositivos, mientras que la segunda particulariza la escucha para ser usada en el modelo energético en cuestión.

A su vez, de forma análoga a lo realizado en el modelo de fuente de energía, se hará uso de la clase auxiliar *LoraRadioEnergyModelHelper*, heredada de la clase base *DeviceEnergyModelHelper,* para la creación y configuración de los modelos de consumos de energía instanciados en los escenarios de simulación. En la figura 27, puede observarse las interfaces públicas de cada una de las clases así como la relación entre ellas.

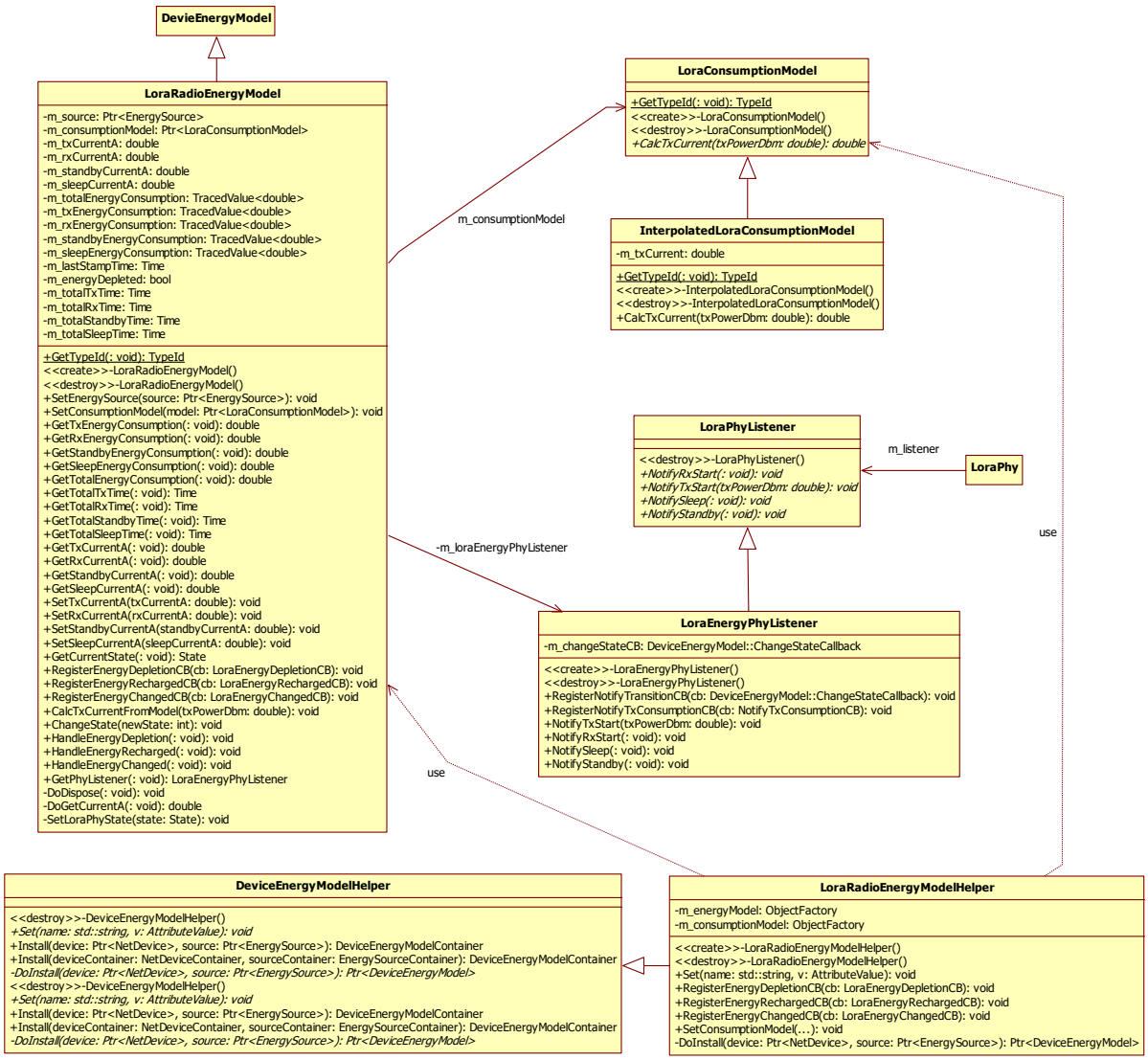


Figura 27. Diagrama de clases modulo de consumo de dispositivos ED.

Establecida la definición de clases e interfaces que caracterizan cada uno de los componentes del modelo de energía, siguiendo las recomendaciones de diseño y arquitectura NS-3 y haciendo uso de su *framework* de energía, queda por determinar el conjunto de acciones a llevar a cabo por los mismo desde que se produce un cambio en el estado de operación de la PHY del dispositivo ED hasta que se actualiza el valor del consumo del dispositivo y el de energía remanente en la fuente de energía. Para ello, cabe distinguir dos procedimientos diferentes, uno correspondiente al caso de una transición del modo de operación a TX y otro al resto de modos de operación, ya que en el primero ha de calcularse el valor de corriente en función de la potencia de operación, mientras que en el resto el consumo es constante según lo indicado en la tabla 6.

En el caso de que se produzca una transición al modo de operación TX, el modelo de PHY del dispositivo de ED, lo notifica al *LoraEnergyPhyListener* y este dispara una petición de cálculo a *InterpolatedConsumptionModel* del valor de corriente correspondiente a la potencia utilizada en la transmisión. Posteriormente el *LoraEnergyPhyListener* notifica el cambio de estado de operación al modelo *LoraRadioEnergyModel*, el cual calcula el valor de consumo del estado anterior, lo añade al cómputo global de consumo y envía una petición al modelo *LoraEnergySource* de actualización del estado de la fuente consultando previamente, el valor de corriente del estado anterior a *LoraRadioEnergyModel* para realizar el cálculo de la energía remanente en la batería y notificar cualquier evento existente en la fuente (cambio del valor de energía, agotamiento o recambio de la fuente) a *LoraRadioEnergyModel.* Por último, *LoraRadioEnergyModel* actualiza su estado a TX, para realizar el cálculo del consumo de éste en el momento que se produzca una nueva transición en el modo de operación.

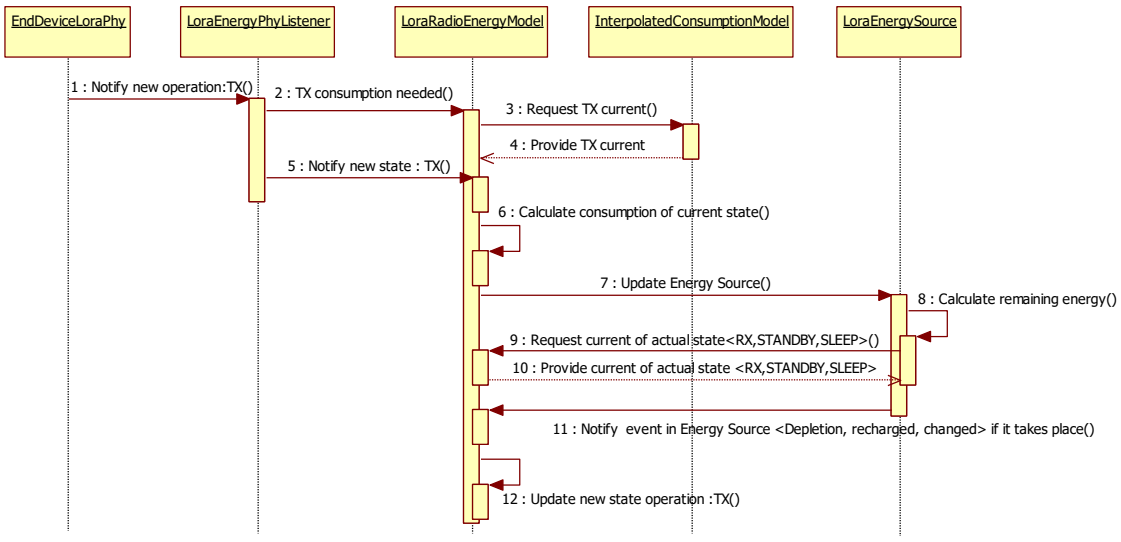


Figura 28. Diagrama secuencial ante transición a modo TX.

En el caso de que se produzca una transición a RX, STANBY o SLEEP, el procedimiento es análogo al anterior, sin embargo, no se producen peticiones de cálculos de corriente desde *LoraRadioEnergyModel a InterpolatedConsumptionModel*, dado que el valor permanece constante durante todo el periodo de simulación según lo indicado en la tabla 6.

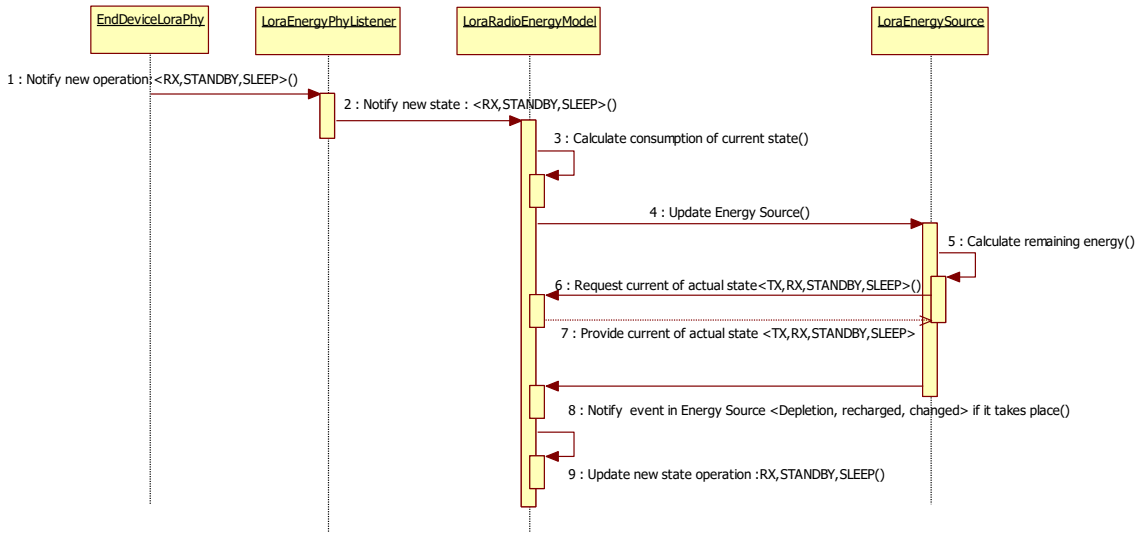


Figura 29. Diagrama secuencial ante transición a modo RX, STANDBY o SLEEP.

5.3.3 Captura de datos y generación de gráficas.

Realizada la caracterización del modelo energético será necesario además implementar mecanismos de generación y recolección de datos durante la simulación de los diferentes escenarios y facilitar la generación de gráficas mediante herramientas como GNUPlot, para llevar a cabo análisis posteriores. Con este fin, se ha implementado la clase auxiliar *LoRaStatsHelper*, la cual provee los siguientes métodos:

- *NodePosition*: Proporciona las coordenadas de cada uno de los nodos (ED y GW) instalados en los escenarios.
- *NodeInformation*: Además de proporcionar información del posicionamiento de los nodos, captura el valor de SF, DR consumo energético y consumo remante hasta el final de la simulación.
- *EnergyInformation*: Provee información detallada del consumo energético de los ED: el valor de tensión de alimentación, el valor de corriente y el tiempo y el consumo energético asociado a cada uno de los modos bajo los que opera los dispositivos ED.
- *Buildings2dInformation / Buildings3dInformation:* Genera un archivo de salida con la localización y dimensiones de los edificios creados en la simulación de escenarios urbanos para su representación en 2d y 3d respectivamente.

- *GnuPlot2dScript / GnuPlot3dScript:* Genera scripts de salida para la creación de gráficas en 2d y 3d respectivamente sobre el consumo de los dispositivos durante el tiempo de simulación.

5.4 Implementación

Implementado la arquitectura y el diseño de software, el presente apartado tiene como objetivo describir las partes más relevantes de la codificación llevada a cabo en cada uno de los componentes que forman parte de la arquitectura del modelo. El código fuente completo asociado a la implementación del presente trabajo se encuentra disponible en [37].

5.4.1 Modelo de fuente de energía

Para particularizar el modelo de fuente de energía descrito en 5.3.2.1, se establecen una serie de atributos en la clase *LoraEnergySource,* con unos valores por defecto iniciales determinados, junto con los métodos de acceso a los mismos para su configuración en caso de ser parametrizables y/o monitorización a lo largo de la simulación. Se ha establecido un valor por defecto de 19980 Julios en el valor de energía, correspondientes a 1500mAh de valor de carga, un valor de 3.7 V, un umbral de agotamiento de batería del 10%, y un ciclo de actualización de 1 segundo. A su vez, se han establecido como valores sujetos a trazabilidad durante el tiempo de simulación los parámetros de energía y carga:

```
static TypeId tid = TypeId ("ns3::LoraEnergySource")
   .SetParent<EnergySource> ()
   .SetGroupName ("Energy")
   .AddConstructor<LoraEnergySource> ()
   .AddAttribute ("LoraEnergySourceInitialEnergyJ",
                  "Initial energy stored in lora energy source.",
                  DoubleValue (19980),
                  MakeDoubleAccessor (&LoraEnergySource::SetInitialEnergy,
                                      &LoraEnergySource::GetInitialEnergy),
                  MakeDoubleChecker<double> ())
   .AddAttribute ("LoraEnergySourceInitialChargemAh",
                  "Initial charge stored in lora energy source (mAh)",
                  DoubleValue (1500),
                  MakeDoubleAccessor (&LoraEnergySource::SetInitialCharge,
                                      &LoraEnergySource::GetInitialCharge),
                  MakeDoubleChecker<double> ())
   .AddAttribute ("LoraEnergySupplyVoltageV",
                  "Initial supply voltage for energy source.",
                  DoubleValue (3.7),
                  MakeDoubleAccessor (&LoraEnergySource::SetSupplyVoltage,
                                      &LoraEnergySource::GetSupplyVoltage),
                  MakeDoubleChecker<double> ())
   .AddAttribute ("LoraEnergyLowBatteryThreshold",
                  "Low battery threshold for energy source.",
                  DoubleValue (0.10),
                  MakeDoubleAccessor (&LoraEnergySource::m_lowBatteryTh),
                  MakeDoubleChecker<double> ())
   .AddAttribute ("LoraEnergyHighBatteryThreshold",
                  "High battery threshold for energy source.",
                  DoubleValue (0.15),
                  MakeDoubleAccessor (&LoraEnergySource::m_highBatteryTh),
                  MakeDoubleChecker<double> ())
   .AddAttribute ("PeriodicEnergyUpdateInterval",
                  "Time between two consecutive periodic energy updates.",
                  TimeValue (Seconds (1.0)),
                  MakeTimeAccessor (&LoraEnergySource::SetEnergyUpdateInterval,
                                    &LoraEnergySource::GetEnergyUpdateInterval),
                  MakeTimeChecker ())
```

```
    .AddTraceSource ("RemainingEnergy",
                     "Remaining energy at LoraEnergySource.",
                   MakeTraceSourceAccessor (&LoraEnergySource::m_remainingEnergyJ),
                     "ns3::TracedValueCallback::Double")
    .AddTraceSource ("RemainingCharge",
                     "Remaining energy at LoraEnergySource.",
                     MakeTraceSourceAccessor (&LoraEnergySource::m_remainingChargemAh),
                     "ns3::TracedValueCallback::Double")
  ;
  return tid;
}
```

Figura 30. Atributos de la clase *LoraEnergySource.*

El cálculo de energía remanente de acuerdo al modelo lineal descrito en 5.3.2, así como el mecanismo de actualización de estado de la batería se ha llevado a cabo a través de los métodos presentes en el modelo *BasicEnergySource* presente en NS-3 [45].

El método *UpdateEnergySource* es el encargado actualizar el estado de la fuente y notificar cualquier evento en la misma al modelo *LoraRadioEnergyModel.* Una vez ejecutado, realiza la llamada del método *CalculateRemaining* para actualizar el valor de energía remanente y disparar los eventos *HandleEnergyDrainedEvent* en caso de que la batería esté agotada, *HandleEnergyRechargedEvent,* en caso de que se produzca una recarga o *NotifyEnergyChanged,* en caso de que el valor de energía haya cambiado con respecto al valor anterior al del proceso de actualización. El método *UpdateEnergySource* es ejecutado ante cualquier cambio en el estado de operación de los dispositivos, tal y como se indica en la figuras 28 y 29, así como cíclicamente según el periodo establecido en el atributo *PeriodicEnergyUpdateInterval*.

```
void
LoraEnergySource::UpdateEnergySource (void)
{
  NS_LOG_FUNCTION (this);
  NS_LOG_DEBUG ("LoraEnergySource:Updating remaining energy.");


  if (Simulator::IsFinished ())
    {
      return;
    }

  m_energyUpdateEvent.Cancel ();

  double remainingEnergy = m_remainingEnergyJ;
  CalculateRemaining();

  m_lastUpdateTime = Simulator::Now ();

  if (!m_depleted && m_remainingEnergyJ <= m_lowBatteryTh * m_initialEnergyJ)
  {
    m_depleted = true;
    HandleEnergyDrainedEvent ();
  }
  else if (m_depleted && m_remainingEnergyJ > m_highBatteryTh * m_initialEnergyJ)
  {
    m_depleted = false;
    HandleEnergyRechargedEvent ();
  }
  else if (m_remainingEnergyJ != remainingEnergy)
  {
    NotifyEnergyChanged ();
  }

  m_energyUpdateEvent = Simulator::Schedule (m_energyUpdateInterval,
                                             &LoraEnergySource::UpdateEnergySource,
                                             this);
}
```

Figura 31. Método de actualización de fuente de energía.

El método *CalculateRemaining* realiza el cálculo de la energía remante en el momento en el que se produce una petición de actualización del estado de la fuente. Para ello, solicita el valor de corriente del conjunto de dispositivos en el estado actual de operación (en el caso del presente trabajo un único dispositivo por fuente) a través del método protegido *CalculateTotalCurrent* de la clase base *EnergySource*, para calcular el valor de energía a decrementar en el cómputo global de energía remanente de acuerdo al modelo descrito en 5.3.2.1.

```
void
LoraEnergySource::CalculateRemaining(void)
{
  NS_LOG_FUNCTION (this);
  double totalCurrentA = CalculateTotalCurrent ();
  Time duration = Simulator::Now () - m_lastUpdateTime;
  NS_ASSERT (duration.IsPositive ());

  double energyToDecreaseJ = (totalCurrentA * m_supplyVoltageV * duration.GetNanoSeconds ()) / 1e9;
  if(m_remainingEnergyJ <= energyToDecreaseJ)
  {
    m_remainingEnergyJ = 0.0;
  }
  m_remainingEnergyJ -= energyToDecreaseJ;
  NS_LOG_DEBUG ("LoraEnergySource:Remaining energy = " << m_remainingEnergyJ);
  m_remainingChargemAh = (m_remainingEnergyJ/m_supplyVoltageV)*1000;
  NS_LOG_DEBUG ("LoraEnergySource:Remaining charge = " << m_remainingChargemAh);
}

/* Base class method */
double
EnergySource::CalculateTotalCurrent (void)
{
  NS_LOG_FUNCTION (this);
  double totalCurrentA = 0.0;
  DeviceEnergyModelContainer::Iterator i;
  for (i = m_models.Begin (); i != m_models.End (); i++)
    {
      totalCurrentA += (*i)->GetCurrentA ();
    }
  return totalCurrentA;
}
```

Figura 32. Método de cálculo de energía remanente y actualización de estado de la fuente (batería).

5.4.2 Modelo de consumo energético del dispositivo

El modelo de consumo de dispositivo está caracterizado por la clase principal *LoraRadioEnergyModel* basada en la clase *WifiRadioEnergyModel* [46] presente en el modelo Wifi de NS-3, y dos clases secundarias, *LoraConsumptionModel* y *LoraEnergyPhyListener*, además de las clases auxiliares, de acuerdo a la arquitectura descrita en 5.3.2.2.

La notificación del cambio de modo de operación del dispositivo ED se lleva a cabo instanciando los métodos *NotifyTxStart, NotifyRxStart, NotifyStandby,* y *NotifySleep* de cada uno de listeners agregados a la PHY del dispositivo ED (en el caso del presente trabajo un único *listener* por PHY) en la implementación de los métodos *SwitchToTx, SwitchToRx, SwitchToStandby* y *SwitchToSleep* de los métodos de la clase *EndDeviceLoraPhy* que caracteriza el modelo PHY del dispositivo ED. En ella se realiza la integración entre el modelo energético del presente trabajo y el *stack* de comunicación de LoraWAN implementado en [4]. Para ello, es necesario añadir los métodos de registro de *LoraPhyListener*, que

utilizará la clase auxiliar *LoraRadioEnergyHelper* para realizar el acoplo tal y como se detalla en 5.4.3.

```
  /* Methods of EndDeviceLoraPhy [4] which instance Notification functions of LoraEnergyPhyListener */

void
EndDeviceLoraPhy::SwitchToTx (double txPowerDbm)
{
  NS_LOG_FUNCTION_NOARGS ();
#if (!ENERGYMODEL_TEST)
  NS_ASSERT (m_state != RX);
#endif
  NS_LOG_DEBUG ("[Phy] Switching to state: " << "TX" << " at time = " << Simulator::Now ().GetSeconds () <<
" s");

  for (Listeners::const_iterator i = m_listeners.begin (); i != m_listeners.end (); i++)
    {
      (*i)->NotifyTxStart (txPowerDbm);
    }

  m_state = TX;
}

void
EndDeviceLoraPhy::SwitchToRx (void)
{
  NS_LOG_FUNCTION_NOARGS ();
#if (!ENERGYMODEL_TEST)
  NS_ASSERT (m_state == STANDBY);
#endif
  NS_LOG_DEBUG ("[Phy] Switching to state: " << "RX" << " at time = " << Simulator::Now ().GetSeconds () <<
" s");

  for (Listeners::const_iterator i = m_listeners.begin (); i != m_listeners.end (); i++)
    {
      (*i)->NotifyRxStart ();
    }

  m_state = RX;
}

void
EndDeviceLoraPhy::SwitchToStandby (void)
{
  NS_LOG_FUNCTION_NOARGS ();
  NS_LOG_DEBUG ("[Phy] Switching to state: " << "STANDBY" << " at time = " << Simulator::Now ().GetSeconds
() << " s");
  for (Listeners::const_iterator i = m_listeners.begin (); i != m_listeners.end (); i++)
    {
      (*i)->NotifyStandby ();
    }

  m_state = STANDBY;
}

void
EndDeviceLoraPhy::SwitchToSleep (void)
{
  NS_LOG_FUNCTION_NOARGS ();
#if (!ENERGYMODEL_TEST)
  NS_ASSERT (m_state == STANDBY);
#endif
  NS_LOG_DEBUG ("[Phy] Switching to state: " << "SLEEP" << " at time = " << Simulator::Now ().GetSeconds ()
<< " s");

  for (Listeners::const_iterator i = m_listeners.begin (); i != m_listeners.end (); i++)
    {
      (*i)->NotifySleep ();
    }

  m_state = SLEEP;
}
```

Figura 33. Notificación cambio de estado de operación desde la clase *EndDeviceLoraPhy* a *LoraEnergyPhyListener*.

Los métodos instanciados anteriores de la clase *LoraEnergyPhyListener*, realizan por su parte la notificación del cambio de estado de la PHY del

dispositivo a la clase *LoraRadioEnergyModel* a través del método privado *m_changeStateCB* que toma como argumento el nuevo estado de operación. En el caso de *NotifyTxStart,* se lleva a cabo además la petición del consumo de corriente a la clase *LoraConsumptionModel* a través del método *m_notifyTxConsumptionCB*, que toma como argumento el valor de potencia usado en la transmisión del dispositivo.

```
void
LoraEnergyPhyListener::NotifyTxStart (double txPowerDbm)
{
  NS_LOG_FUNCTION (this << txPowerDbm);
  NS_LOG_DEBUG ("[Listener] Notify new state: " << "TX" << " at time = " << Simulator::Now ().GetSeconds ()
<< " s");

  //Update  Tx consumption
  NS_ASSERT (!m_notifyTxConsumptionCB.IsNull ());
  m_notifyTxConsumptionCB (txPowerDbm);

  NS_ASSERT (!m_changeStateCB.IsNull ());
  m_changeStateCB (EndDeviceLoraPhy::TX);

}

void
LoraEnergyPhyListener::NotifyRxStart ()
{
  NS_LOG_FUNCTION (this);
  NS_LOG_DEBUG ("[Listener] Notify new state: " << "RX" << " at time = " << Simulator::Now ().GetSeconds ()
<< " s");
  NS_ASSERT (!m_changeStateCB.IsNull ());
  m_changeStateCB (EndDeviceLoraPhy::RX);
}

void
LoraEnergyPhyListener::NotifyStandby (void)
{
  NS_LOG_FUNCTION (this);
  NS_LOG_DEBUG ("[Listener] Notify new state: " << "STANDBY" << " at time = " << Simulator::Now
().GetSeconds () << " s");
  NS_ASSERT (!m_changeStateCB.IsNull ());
  m_changeStateCB (EndDeviceLoraPhy::STANDBY);
}

void
LoraEnergyPhyListener::NotifySleep (void)
{
  NS_LOG_FUNCTION (this);
  NS_LOG_DEBUG ("[Listener] Notify new state: " << "SLEEP" << " at time = " << Simulator::Now ().GetSeconds
() << " s");
  NS_ASSERT (!m_changeStateCB.IsNull ());
  m_changeStateCB (EndDeviceLoraPhy::SLEEP);
}
```

Figura 34. Implementación de notificadores de cambio de estado de *LoraEnergyPhyListener*.

El cálculo del valor de corriente asociado a una potencia de transmisión determinada se lleva a cabo a través del método *CalcTxCurrent* de la clase *InterpolatedLoraConsumptionModel,* el cual es instanciado indirectamente en el método *NotifyTxStart* de *LoraEnergyPhyListener* a través del método privado *m_notifyTxConsumptionCB* que registra como *callback* al método *CalcTxCurrentFromModel* de *LoraRadioEnergyModel* y éste instancia a su vez a *CalcTxCurrent.* En él, se lleva a cabo la codificación del método de interpolación de corriente de la fórmula 5.4 de 5.3.2.2 a partir de los valores de consumo de la hoja de datos de los dispositivos ED Semtech SX1272 [7].

```
double
InterpolatedLoraConsumptionModel::CalcTxCurrent (double power_dBm) const
{
  NS_LOG_FUNCTION (this << power_dBm);
  //Collect data from datasheet
  std::vector<double> power_dBm_lookup_table =      {7.0, 13.0, 17.0, 20.0 };
  std::vector<double> curruent_ma_lookup_table =   {18.0, 28.0, 90.0, 125.0};

  //Size of look up table elements
  int n_elements = power_dBm_lookup_table.size();

  //Values which limits the value to be interpolated
  double current_ma_L, current_ma_R, power_dBm_L, power_dBm_R;

  //Value of current result of interpolation
  double current_ma_interpolated;

  //Index
  int index = 0;

  NS_ASSERT ( power_dBm < power_dBm_lookup_table[n_elements - 1] || power_dBm  > power_dBm_lookup_table[0]
);

  while ( power_dBm >power_dBm_lookup_table[index +1] )
  {
    index++;
  }

  //Prepare terms of the formula (For more reference : “Integración de un modelo de energía en la
simulación de redes LoRaWAN en NS-3 ”, section 5.3.2.2)
  current_ma_L = curruent_ma_lookup_table[index];
  power_dBm_L  =  power_dBm_lookup_table[index];

  current_ma_R =curruent_ma_lookup_table[index +1];
  power_dBm_R  =  power_dBm_lookup_table[index +1];

  current_ma_interpolated = current_ma_L + (current_ma_R - current_ma_L)/(power_dBm_R - power_dBm_L) *
(power_dBm - power_dBm_L);

  NS_LOG_DEBUG ("Input Power: "<< power_dBm << "dBm - Interpolated Current: " << current_ma_interpolated <<
" mA");

  return (current_ma_interpolated / 1000);

}
```

Figura 35. Método de cálculo de corriente en modo TX basado en interpolación.

Para particularizar el modelo de consumo de dispositivo descrito en 5.3.2.2, se establecen una serie de atributos en la clase *LoraRadioEnergyModel,* con unos valores por defecto iniciales determinados, junto con los métodos de acceso a los mismos para su configuración en caso de ser parametrizables y/o monitorización a lo largo de la simulación. Se ha establecido unos valores de corriente en los modos RX, STANDBY y SLEEP de 11.2 mA, 1.4 mA y 1.8 μA respectivamente. A su vez se ha establecido un valor por defecto de 43.5 mA en el modo TX, que será modificado en el momento en el que se instancie en la simulación el modelo de interpolación de corriente en función de la potencia de transmisión utilizada. Junto con los valores de corriente, en la lista de atributos se encuentra el miembro privado *m_consumptionModel* como puntero que señala al modelo de consumo utilizado en la simulación (en el caso del presente trabajo *InterpolatedLoraConsumptionModel*). Además, se han establecido como valores sujetos a trazabilidad durante el tiempo de simulación el valor de consumo de corriente en cada uno de los modos de operación de los dispositivos ED, así como el cómputo global.

```
 TypeId
LoraRadioEnergyModel::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::LoraRadioEnergyModel")
    .SetParent<DeviceEnergyModel> ()
    .SetGroupName ("Energy")
    .AddConstructor<LoraRadioEnergyModel> ()
   //Default atributes (For more reference : "Integración de un modelo de energía en la simulación de redes
LoRaWAN en NS-3 ", section 5.3.2.2)
    .AddAttribute  ("TxCurrentA",
                    "Default consumption in TX operation)",
                    DoubleValue (TX_CURR_DEFAULT),
                    MakeDoubleAccessor (&LoraRadioEnergyModel::SetTxCurrentA,
                                        &LoraRadioEnergyModel::GetTxCurrentA),
                    MakeDoubleChecker<double> ())
    .AddAttribute  ("RxCurrentA",
                    "Default consumption in RX operation)",
                    DoubleValue (RX_CURR_DEFAULT),
                    MakeDoubleAccessor (&LoraRadioEnergyModel::SetRxCurrentA,
                                        &LoraRadioEnergyModel::GetRxCurrentA),
                    MakeDoubleChecker<double> ())
    .AddAttribute  ("StandbyCurrentA",
                    "Default consumption in STANDBY operation)",
                    DoubleValue (STANDBY_CURR_DEFAULT),
                    MakeDoubleAccessor (&LoraRadioEnergyModel::SetStandbyCurrentA,
                                        &LoraRadioEnergyModel::GetStandbyCurrentA),
                    MakeDoubleChecker<double> ())
    .AddAttribute  ("SleepCurrentA",
                    "Default consumption in SLEEP operation)",
                    DoubleValue (SLEEP_CURR_DEFAULT),
                    MakeDoubleAccessor (&LoraRadioEnergyModel::SetSleepCurrentA,
                                        &LoraRadioEnergyModel::GetSleepCurrentA),
                    MakeDoubleChecker<double> ())
    .AddAttribute  ("ConsumptionModel", "A pointer to the attached consumption model.",
                    PointerValue (),
                    MakePointerAccessor (&LoraRadioEnergyModel::m_consumptionModel),
                    MakePointerChecker<LoraConsumptionModel> ())
    .AddTraceSource("TotalEnergyConsumption",
                     "Total energy consumption of the radio device.",
                     MakeTraceSourceAccessor (&LoraRadioEnergyModel::m_totalEnergyConsumption),
                    "ns3::TracedValueCallback::Double")
    .AddTraceSource("TxEnergyConsumption",
                     "Energy consumption in TX mode",
                     MakeTraceSourceAccessor (&LoraRadioEnergyModel::m_txEnergyConsumption),
                     "ns3::TracedValueCallback::Double")
    .AddTraceSource("RxEnergyConsumption",
                    "Energy consumption in RX mode",
                     MakeTraceSourceAccessor (&LoraRadioEnergyModel::m_rxEnergyConsumption),
                    "ns3::TracedValueCallback::Double")
    .AddTraceSource("StandbyEnergyConsumption",
                    "Energy consumption in STANDBY mode",
                    MakeTraceSourceAccessor (&LoraRadioEnergyModel::m_standbyEnergyConsumption),
                    "ns3::TracedValueCallback::Double")
    .AddTraceSource("SleepEnergyConsumption",
                    "Energy consumption in SLEEP mode",
                    MakeTraceSourceAccessor (&LoraRadioEnergyModel::m_sleepEnergyConsumption),
                    "ns3::TracedValueCallback::Double")

  ;
  return tid;
}
```

Figura 36. Atributos de la clase *LoraRadioEnergyModel.*

El método *ChangeState* de *LoraRadioEnergyModel* es utilizado como función *callback* del método de notificación de cambio de estado *m_changeStateCB* de *LoraEnergyPhyListener*. En él se realiza el cálculo del consumo correspondiente al estado de operación previo a la transición hacia el nuevo estado de acuerdo a la ecuación 5.2 y se añade al cómputo global del consumo del dispositivo ED durante la simulación almacenado en la variable privada *m_totalEnergyConsumption*. Tras el cálculo, se instancia el método *UpdateEnergySource* del modelo de fuente de energía (en el caso del presente trabajo *LoraEnergySource)* referenciado a través del miembro *m_source* de *LoraRadioEnergyModel,* y posteriormente se notifica la transición hacia el nuevo

estado a través de la llamada del método *SetLoraPhyState* para referenciar al nuevo estado de operación en el cálculo de consumo ante una nueva transición de estado.

```
// Implementation based on WiFi model (already tested in platform). Adaptation for Lora-PHY
void
LoraRadioEnergyModel::ChangeState (int newState)
{
  NS_LOG_FUNCTION (this << newState);
  Time duration = Simulator::Now () - m_lastStampTime;

  //Variable to handle energy decrement
  double energyDecrement = 0.0;
  double supplyVoltage = m_source->GetSupplyVoltage ();

  //Calculate Energy and register Time elapsed and Energy consumed in each operation state
  switch (m_currentState)
    {
    case EndDeviceLoraPhy::TX:
      energyDecrement = duration.GetSeconds () * m_txCurrentA * supplyVoltage;
      m_totalTxTime += duration;
      m_txEnergyConsumption += energyDecrement;
      break;
    case EndDeviceLoraPhy::RX:
      energyDecrement = duration.GetSeconds () * m_rxCurrentA * supplyVoltage;
      m_totalRxTime += duration;
      m_rxEnergyConsumption += energyDecrement;
      break;
    case EndDeviceLoraPhy::STANDBY:
      energyDecrement = duration.GetSeconds () * m_standbyCurrentA * supplyVoltage;
      m_totalStandbyTime += duration;
      m_standbyEnergyConsumption += energyDecrement;
      break;
    case EndDeviceLoraPhy::SLEEP:
      energyDecrement = duration.GetSeconds () * m_sleepCurrentA * supplyVoltage;
      m_totalSleepTime += duration;
      m_sleepEnergyConsumption += energyDecrement;
      break;
    default:
      NS_FATAL_ERROR ("Invalid Lora operation State: " << m_currentState);
    }
  // update total energy consumption
  m_totalEnergyConsumption += energyDecrement;
  // update last update time stamp
  m_lastStampTime = Simulator::Now ();
  // notify energy source
  m_source->UpdateEnergySource ();
  //If energy not depleted, change state and inform about energy consumption
  if (m_energyDepleted == false)
    {
      SetLoraPhyState (static_cast<EndDeviceLoraPhy::State>(newState));
      NS_LOG_INFO ("Energy consumption is " << m_totalEnergyConsumption << "J");
    }
}

void
LoraRadioEnergyModel::SetLoraPhyState (const EndDeviceLoraPhy::State state)
{
  NS_LOG_FUNCTION (this << state);
  m_currentState = state;
  std::string stateName;
  switch (state)
    {
    case EndDeviceLoraPhy::STANDBY:
      stateName = "STANDBY";
      break;
    case EndDeviceLoraPhy::TX:
      stateName = "TX";
      break;
    case EndDeviceLoraPhy::RX:
      stateName = "RX";
      break;
    case EndDeviceLoraPhy::SLEEP:
      stateName = "SLEEP";
      break;
    }
    NS_LOG_DEBUG ("[EnergyModel] Switching to state: " << stateName << " at time = "
    << Simulator::Now   ().GetSeconds () << " s");
}
```

Figura 37. Proceso de cálculo de consumo ante cambio de estado de operación del dispositivo ED.

5.4.3 Clases auxiliares (“helpers”)

Para facilitar la configuración de parámetros, la monitorización y captura de datos durante la simulación de escenarios donde se aplica el modelo de energía del presente trabajo, se ha hecho uso de tres clases auxiliares, *LoraEnergySourceHelper, LoraRadioEnergyModelHelper* y *LoraStatsHelper.*

LoraEnergySourceHelper es una clase derivada de la clase base *EnergySourceHelper* perteneciente al *framework* de energía en NS-3, mediante la cual pueden configurarse cada uno de los atributos de la clase *LoraEnergySource* haciendo uso del método *Set,* y a su vez llevar a cabo la instalación del modelo de fuente de energía durante la simulación haciendo uso del método *Install* perteneciente a la clase base que instancia a su vez el método *DoInstall* de la clase derivada.

```
void
LoraEnergySourceHelper::Set (std::string name, const AttributeValue &v)
{
  m_loraEnergySource.Set (name, v);
}


Ptr<EnergySource>
LoraEnergySourceHelper::DoInstall (Ptr<Node> node) const
{
  NS_ASSERT (node != NULL);
  Ptr<EnergySource> source = m_loraEnergySource.Create<LoraEnergySource> ();
  NS_ASSERT (source != NULL);
  source->SetNode (node);
  return source;
}
```

Figura 38. Métodos de *LoraEnergySourceHelper* para la configuración e instalación de *LoraEnergySource*.

De forma análoga, la clase *LoraRadioEnergyModelHelper* derivada de *DeviceEnergyModelHelper* y basada en *WifiRadioEnergyModelHelper* [46], dispone del método *Set* para configurar los atributos de *LoraRadioEnergyModel,* del método *SetConsumptionModel* para agregar el modelo de consumo *InterpolatedLoraConsumptionModel* y del método *DoInstall,* instanciado por el método *Install* de la clase base, para llevar a cabo la instalación del modelo de consumo de energía del dispositivo ED durante la simulación. En este último, se agrega el modelo de fuente *LoraEnergySource* al dispositivo de red *Lora*, se registra el *LoraEnergyPhyListener* en la clase PHY para realizar la monitorización, así como las funciones *callbacks* que pueden ser utilizadas en la simulación para ser instanciadas ante cualquier evento reportado por la fuente de energía.

```
void
LoraRadioEnergyModelHelper::Set (std::string name, const AttributeValue &v)
{
  m_energyModel.Set (name, v);
}

void
LoraRadioEnergyModelHelper::SetConsumptionModel (std::string name)
{
  ObjectFactory factory;
  factory.SetTypeId (name);
  m_consumptionModel = factory;
}
```

```
Ptr<DeviceEnergyModel>
LoraRadioEnergyModelHelper::DoInstall (Ptr<NetDevice> device,
                                       Ptr<EnergySource> source) const
{
  NS_ASSERT (device != NULL);
  NS_ASSERT (source != NULL);
  //Check correct device
  std::string deviceName = device->GetInstanceTypeId ().GetName ();
  if (deviceName.compare ("ns3::LoraNetDevice") != 0)
    {
      NS_FATAL_ERROR ("NetDevice type is not LoraNetDevice!");
    }
  //SetEnergy Source and add model in energy source
  Ptr<Node> node = device->GetNode ();
  Ptr<LoraRadioEnergyModel> model = m_energyModel.Create ()->GetObject<LoraRadioEnergyModel> ();
  NS_ASSERT (model != NULL);
  model->SetEnergySource (source);
  source->AppendDeviceEnergyModel (model);

  //Register PhyListener
  Ptr<LoraNetDevice> loraDevice = device->GetObject<LoraNetDevice> ();
  Ptr<EndDeviceLoraPhy> loraPhy = loraDevice->GetPhy ()->GetObject<EndDeviceLoraPhy> ();
  loraPhy->RegisterListener (model->GetPhyListener ());

  //Register Energy-handling callbacks
  if (!m_energyDepletionCB.IsNull())
    {
      model->RegisterEnergyDepletionCB(m_energyDepletionCB);
    }
  if (!m_energyRechargedCB.IsNull())
    {
      model->RegisterEnergyRechargedCB(m_energyRechargedCB);
    }
  if (!m_energyChangedCB.IsNull())
    {
      model->RegisterEnergyChangedCB(m_energyChangedCB);
    }

  //Set Consumption Model
  if (m_consumptionModel.GetTypeId ().GetUid ())
    {
      Ptr<LoraConsumptionModel> consumption = m_consumptionModel.Create<LoraConsumptionModel> ();
      model->SetConsumptionModel (consumption);
    }
  return model;
}
```

Figura 39. Métodos de *LoraRadioEnergyModelHelper* para la configuración e instalación de *LoraRadioEnergyModel.*

A través de la clase *LoraStatsHelper* se lleva a cabo la captura de datos durante la simulación para su posterior análisis. En la figura 40 puede observarse el procedimiento de captura del método *NodeInformation* utilizado en los escenarios simulados en 6.3.

NodeInformation captura los datos de posicionamiento, el valor de DR y SF, y el consumo de energía de los dispositivos ED, así como el posicionamiento de los GW utilizados en la simulación, aprovechando los mecanismos de agregación de objetos de NS-3, a partir de los cuales se capturan los diferentes modelos asociados a cada uno de los nodos que están presentes en la simulación, almacenados en los contenedores *NodeContainer.* La captura de datos de posicionamiento se lleva a cabo mediante el uso de clase *MobilityModel* de NS-3 y su método *GetPosition,* la recopilación de datos del consumo energético se realiza mediante el uso de los métodos *GetRemainingEnergy* de *LoraEnergySource* y *GetTotalEnergyConsumption* de *LoraRadioEnergyModel* , y los valores de DR y SF se obtienen a través de los métodos *GetDataRate y GetSfFromDataRate* respectivamente de la clase *EndDeviceLoraMac* perteneciente al módulo de LoRaWAN en [4].

Cada uno de los parámetros son guardados en un archivo de salida que será utilizado como registro de entrada junto con otros datos generados por otros métodos de la clase, tales como *Buildings2dInformation, Buildings3dInformation,* por los scripts *GnuPlot* generados en los métodos *GnuPlot2dScript* y *GnuPlot3dScript*, para la generación de gráficas de los escenarios simulados.

```
void LoraStatsHelper::NodeInformation (std::string fileName, NodeContainer endDevices, NodeContainer
gateways)
{
  const char * name = fileName.c_str();
  std::ofstream nodeInformationFile;
  nodeInformationFile.open(name);
  NS_ASSERT(nodeInformationFile.is_open() == true);

  NS_LOG_DEBUG ("Collecting Node Information");
  //Print column info
  nodeInformationFile <<"#Dev"<<" "<<"nodeId"<<" " <<"x"<<" " <<"y" <<" "<<"z" << " "
                      << "SF" <<" "<< "DR"   << " "<<"ConsEnergy"<< " "<< "RemEnergy" << " " << std::endl;

  //End Devices Information
  for (NodeContainer::Iterator i = endDevices.Begin(); i!=endDevices.End(); i++)
    {
      Ptr<Node> node = *i;
      uint nodeId = node->GetId();

      //Get mobility info
      Ptr<MobilityModel> mobility = node->GetObject<MobilityModel>();
      NS_ASSERT (mobility != NULL);
      Vector position =  mobility->GetPosition ();

      //Get energy info
      Ptr<EnergySourceContainer> energySourceContainer = node->GetObject<EnergySourceContainer>();
      NS_ASSERT (energySourceContainer != NULL);
      Ptr<LoraEnergySource> loraEnergySource =
      DynamicCast<LoraEnergySource>(energySourceContainer->Get(0));
      NS_ASSERT (loraEnergySource != NULL);
      double remainingEnergyJ = loraEnergySource->GetRemainingEnergy();
      DeviceEnergyModelContainer =
      loraEnergySource->FindDeviceEnergyModels("ns3::LoraRadioEnergyModel");
      Ptr<LoraRadioEnergyModel> loraRadioEnergyModel =
      DynamicCast<LoraRadioEnergyModel>(deviceEnergyModelContainer.Get(0));
      NS_ASSERT (loraRadioEnergyModel != NULL);
      double consumedEnergyJ   = loraRadioEnergyModel->GetTotalEnergyConsumption();

      //Get lora-protocol info
      Ptr<NetDevice> netDevice = node->GetDevice(0);
      NS_ASSERT(netDevice != NULL);
      Ptr<LoraNetDevice> loraNetDevice = netDevice->GetObject<LoraNetDevice>();
      NS_ASSERT(loraNetDevice != NULL);
      Ptr<EndDeviceLoraMac> edMac= loraNetDevice->GetMac()->GetObject<EndDeviceLoraMac>();
      NS_ASSERT(edMac != NULL);
      uint  dataRate = edMac->GetDataRate();
      uint  spreadingFactor = edMac->GetSfFromDataRate(dataRate);

      //Print Info
      nodeInformationFile <<"ED" <<" "<<nodeId<<" "<<position.x<<" "<<position.y<< " "
                          <<position.z<< " "<<spreadingFactor<<" "<<dataRate<< " "
                          <<consumedEnergyJ<<" "<<remainingEnergyJ<< " "<< std::endl;
    }

  // Gateways Information
  for (NodeContainer::Iterator i = gateways.Begin (); i != gateways.End (); ++i)
    {
      Ptr<Node> node = *i;
      uint nodeId = node->GetId();

      //Get mobility info
      Ptr<MobilityModel> mobility = node->GetObject<MobilityModel>();
      NS_ASSERT (mobility != NULL);
      Vector position =  mobility->GetPosition ();
      //Print Info
      nodeInformationFile <<"GW"<<" "<<nodeId<<" "<< position.x<< " "<<position.y<<" "
                          << position.z <<" "<< std::endl;

    }
}
```

Figura 40. Ejemplo de captura de datos utilizado en el método *NodeInformation* de la clase *LoraStatsHelper.*

El resto de los métodos de la clase *LoraStatsHelper* hacen uso de mecanismos análogos a los utilizados en *NodeInformation* para realizar la captura de datos y generación y/o generación de *scripts* GNUPlot. Todos ellos se encuentran disponibles en el repositorio [37].

5.5 Tests y validación

Para llevar a cabo la validación del modelo de energía implementado se ha diseñado un *test-suite* llamado *lora-radio-energy-model* disponible en el repositorio [37] con el resto de los componentes de software que forman parte de la implementación del modelo en sí. Es importante señalar que el protocolo valida cada uno de los componentes que forman parte del modelo de energía descrito en 5.3, no el resto de los componentes asociados al *stack* presentes en el repositorio base [4], cuyos modelos poseen sus propios protocolos de validación.

El *test-suite,* con una duración total de 5.5 s, se basa en un escenario de un único dispositivo final ED con un *stack* de LoRaWAN configurado, presente en el conjunto de escenarios que se detallan en 6.4, al cual se le agrega el modelo de energía sujeto a estudio, y en el cual se introducen los siguientes datos de operación:

- Tensión de alimentación de la batería: 3.7 V
- Energía inicial de la batería: 5.5 J (reducido valor consecuente con la resolución del test)
- Potencia de transmisión del dispositivo (TX) 14 dBm.
- Consumo de corriente: RX-11.2 mA, STANDBY-1.4 mA, SLEEP – 1.8 μA

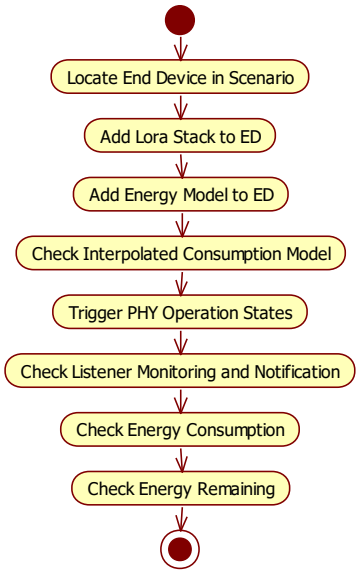


Figura 41. Protocolo de validación modelo de Energía.

Configurado el modelo, el test presenta una serie de eventos de entrada que estimulan el módulo de energía presente en el dispositivo, generando con ello una serie de salidas que son utilizadas para contrastar la información esperada con la obtenida en el protocolo de validación (figura 41). La fase de estímulo se

encarga de obtener información acerca del modelo de cálculo del consumo de corriente llevada a cabo por *InterpolatedLoraConsumptionModel*, de la correcta monitorización de los estados de operación realizada por *LoraRadioEnergyModelListener*, del cálculo de consumo energético en cada uno de los modos de operación efectuada por *LoraRadioEnergyModel* y de la correcta monitorización de los niveles de energía por parte de *LoraEnergySource.* La figura 42 muestra parte del código implementado en el *test-suite,* donde puede observarse la fase de captura de datos, y el disparo de los diferentes eventos de entrada que estimulan el modelo de energía.

```
/**************************************************************
 *  Get Test information
 **************************************************************/
//Get Phy Information
Ptr<Node> node = endDevices.Get(0);
NS_ASSERT (node != NULL);
Ptr<NetDevice> netDevice = node->GetDevice(0);
NS_ASSERT (netDevice != NULL);
Ptr<LoraNetDevice> loraNetDevice = netDevice->GetObject<LoraNetDevice>();
NS_ASSERT(loraNetDevice != NULL);
Ptr<EndDeviceLoraPhy> edPhy = loraNetDevice->GetPhy()->GetObject<EndDeviceLoraPhy>();
NS_ASSERT(edPhy!= NULL);

//Get Energy and consumption Information
Ptr<EnergySourceContainer> energySourceContainer = node->GetObject<EnergySourceContainer>();
NS_ASSERT (energySourceContainer != NULL);
Ptr<LoraEnergySource> loraEnergySource = DynamicCast<LoraEnergySource>(energySourceContainer->Get(0));
NS_ASSERT (loraEnergySource != NULL);
DeviceEnergyModelContainer = loraEnergySource->FindDeviceEnergyModels("ns3::LoraRadioEnergyModel");
Ptr<LoraRadioEnergyModel> loraRadioEnergyModel = DynamicCast<LoraRadioEnergyModel>(deviceEnergyModelContainer.Get(0));
NS_ASSERT (loraRadioEnergyModel != NULL);

/**************************************************************
 *  Test Interpolated Consumption Model
 **************************************************************/
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 7.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 8.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 9.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 10.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 11.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 12.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 13.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 14.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 15.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 16.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 17.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 18.0 );
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 19.0);
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &LoraRadioEnergyModel::CalcTxCurrentFromModel, loraRadioEnergyModel, 20.0);

/**************************************************************
 *  Test Listener
 **************************************************************/
Simulator::Schedule (Seconds(START_SIMULATION_TIME), &EndDeviceLoraPhy::SwitchToTx,edPhy, TX_POWER_DEFAULT);
Simulator::Schedule (Seconds(1), &EndDeviceLoraPhy::SwitchToRx, edPhy);
Simulator::Schedule (Seconds(2.25), &EndDeviceLoraPhy::SwitchToStandby, edPhy);
Simulator::Schedule (Seconds(3.75), &EndDeviceLoraPhy::SwitchToSleep, edPhy);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &EndDeviceLoraPhy::SwitchToStandby, edPhy);

/**************************************************************
 *  Test Energy Device Model
 **************************************************************/
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetTxCurrentA, loraRadioEnergyModel);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetRxCurrentA, loraRadioEnergyModel);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetStandbyCurrentA, loraRadioEnergyModel) ;
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetSleepCurrentA, loraRadioEnergyModel);

Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetTotalTxTime, loraRadioEnergyModel);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetTotalRxTime, loraRadioEnergyModel) ;
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetTotalStandbyTime, loraRadioEnergyModel);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetTotalSleepTime, loraRadioEnergyModel);

Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetTxEnergyConsumption, loraRadioEnergyModel);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetRxEnergyConsumption, loraRadioEnergyModel) ;
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetStandbyEnergyConsumption, loraRadioEnergyModel);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetSleepEnergyConsumption, loraRadioEnergyModel);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraRadioEnergyModel::GetTotalEnergyConsumption, loraRadioEnergyModel);

/**************************************************************
 *  Test Energy Source Model
 **************************************************************/
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraEnergySource::GetSupplyVoltage, loraEnergySource);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraEnergySource::GetInitialEnergy, loraEnergySource);
Simulator::Schedule (Seconds(STOP_SIMULATION_TIME), &LoraEnergySource::GetRemainingEnergy, loraEnergySource);
```

Figura 42. Captura de datos y disparo de eventos en el test-suite del modelo de energía.

Para comprobar la validez del modelo de consumo de corriente de transmisión basado en interpolación potencia-corriente, se dispone de una serie de valores de entrada de potencia de transmisión en el rango descrito en 5.3.3, que generan una serie de valores de salida, los cuales son comparados con los valores esperados. Cualquier valor fuera del rango genera un error en la simulación previo a la ejecución del escenario. En la tabla 7, se presenta los valores de entrada y los resultados obtenidos.

Potencia de entrada	Valor de corriente esperado	Valor de corriente de salida
7 dBm	18 mA	18 mA
8 dBm	19.6667 mA	19.6667 mA
9 dBm	21.3333 mA	21.3333 mA
10 dBm	23 mA	23 mA
11 dBm	24.6667 mA	24.6667 mA
12 dBm	26.3333 mA	26.3333 mA
13 dBm	28 mA	28 mA
14 dBm	43.5 mA	43.5 mA
15 dBm	59 mA	59 mA
16 dBm	74.5 mA	74.5 mA
17 dBm	90 mA	90 mA
18 dBm	101.667 mA	101.667 mA
19 dBm	113.333 mA	113.333 mA
20 dBm	43.5 mA	43.5 mA

Tabla 7. Resultados del protocolo de validación del modelo consumo de corriente interpolado.

La validación del proceso de escucha de los estados de operación está basada en la programación y disparo en tiempos discretos de transiciones en los cambios de operación (TX, RX, STANDBY, SLEEP), del transceptor de radio (capa PHY) presente en [4], al cual se adjuntan los *LoraRadioEnergyModelPhyListener* para monitorizar el estado de operación de los ED y notificar cambios de estado al módulo *LoraRadioEnergyModel*. Disparadas las transiciones, se comprueba la línea temporal de la cadena de notificación desde el transceptor de radio hasta el módulo de energía. Por defecto, los disparos del simulador presente en el *test-suite* se llevan a cabo en los tiempos discretos 0.0s, 1.0s, 2.25s y 3.75s, que serán utilizados a su vez para comprobar el consumo de energía del dispositivo de acuerdo a los diferentes modos de operación. Conviene señalar que la secuenciación de las transiciones de los cambios de operación no sigue la lógica presente en el modelo de PHY en [4], si no que se establecen unos intervalos temporales que facilitan la medida de los datos obtenidos. El protocolo es responsable de la comprobación de la correcta monitorización y notificación de los estados de funcionamiento de la capa física. La verificación de la correcta secuenciación de los estados de operación de la capa PHY de LoRa es llevada a cabo en un *test-suite* específico presente en [4], responsable de su validación. Dada la independencia de ambos modelos de software, la validación de acuerdo a la secuencia anterior planteada implica indirectamente la validez del modelo energético en cualquier condición de operación de la capa física PHY, incluyendo la presente en [4] e incluso en futuras implementaciones que contemplen redes LoRaWAN con dispositivos de clase B y C.

Estado disparado	Disparo	Notificación a *Listener*	Notificación a modelo de energía
TX	0.0000 s	0.0000 s	0.0000 s
RX	1.0000 s	1.0000 s	1.0000 s
STANDBY	2.2500 s	2.2500 s	2.2500 s
SLEEP	3.7500 s	3.7500 s	3.7500 s
FIN SIMULACIÓN (Transición a STANDBY)	5.5000 s	5.5000 s	5.5000 s

Tabla 8.Resultados del protocolo de validación de la monitorización y reporte de los estados de operación.

Tal y como se observa en la Tabla 8, el proceso de notificación es tan rápido que ni si quiera se aprecia la separación temporal entre la ejecución de los diferentes procesos de la cadena debido a la eficiencia computacional del simulador en los equipos de usuario de la generación actual.

Para validar el modelo de cálculo de energía del dispositivo, el protocolo de test hace uso de los procesos de disparo anteriores, que establecen cuatro intervalos diferentes de operación del dispositivo: TX – 1.0 s, RX – 1.25 s, STANDBY – 1. 5 y SLEEP – 1.75 s, y del consumo de corriente capturado en cada uno de ellos (que deberá ser coherente con los datos introducidos en el protocolo), para llevar a cabo el cálculo del consumo energético en Julios en cada uno de los estados, así como el consumo total durante el tiempo de simulación (5.5 s). En la Tabla 10, pueden apreciarse los resultados obtenidos.

Estado	Consumo de corriente capturado	Tensión de alimentación capturada	Tiempo de operación	Consumo energético esperado	Consumo energético esperado de salida
TX	43.5 mA	3.7 V	1.0000 s	0.1609 J	0.1609 J
RX	112 mA		1.250 s	0.0518 J	0.0518 J
STANDBY	112 mA		2.2500 s	0.0077 s	0.0077 J
SLEEP	1.8 μA		3.7500 s	1.1655 10^{-5} J	1.1655 10^{-5}J
OPERACIÓN CONJUNTA				0.2205 J	0.220532 J

Tabla 9.Resultados del protocolo de validación del consumo energético del dispositivo

Por último, a partir de los datos generados en el proceso anterior se comprueba la validez del modelo de fuente de energía contrastando la energía restante en la fuente adjunta al dispositivo, haciendo uso de la captura de energía inicial introducida como dato en el protocolo de validación:

Energía inicial introducida	Energía inicial capturada	Consumo de energía	Energía restante esperada	Energía restante de salida
5.55 J	5.55 J	0.2253 J	5.329468 s	5.329468 s

Tabla 10. Resultados del protocolo de validación del modelo de fuente de energía.

La validación del modelo energético a través del protocolo anterior garantiza la consistencia de los datos obtenidos en los escenarios de operación planteados en 5.6, de acuerdo a las métricas definidas y los resultados obtenidos tras la

variación de las condiciones y parámetros de operación en cada uno de los mismos.

5.6 Escenarios

Para llevar a cabo la simulación del modelo de energía implementado en el presente trabajo, se han modelado dos escenarios principales relacionados con aplicaciones y entornos de interés en el contexto de IoT en el que se despliegan un número variable de ED de clase A y GW. El primero está basado en un área de extensión abierta en el cual existe visión directa (LOS) entre los dispositivos, inspirado en la monitorización y envío de datos en el ámbito de la agricultura bajo una red de sensores, tales como la luminosidad, la humedad, la temperatura, el caudal de riego, etc., donde se contempla un alcance de los dispositivos de hasta 10Km, tal y como contempla la especificación. El segundo está basado en un núcleo urbano donde no existe visión directa (NLOS) entre los dispositivos, inspirado en la monitorización y envío de datos en el ámbito de las *Smart Cities,* y de las *Smart Homes,* tales como el nivel de contaminación, la luminosidad, el consumo energético, el consumo de gas, etc., donde se contempla un alcance de los dispositivos de hasta 2Km, tal y como contempla la literatura actual. En ambos escenarios los ED representan la red sensorial de recogida y envío de datos, en los cuales se instala el modelo de energía, dado su típico modo de funcionamiento autónomo, y los GW representan los nodos receptores que de los mensajes capturados por los sensores. En el despliegue de ambos escenarios se ha considerado únicamente el envío en sentido ascendente del enlace (UP) de la red LoRaWAN, es decir, el envío de los datos recolectados por los EDs hacia el GW, el cual realizaría un envío hipotético a un servidor central NS en un escenario real para llevar a cabo la recolección y análisis de datos, sin considerar con ello la transmisión en sentido descendente (DL). Esto implica que la asignación de los SF bajo los cuales operan los dispositivos se lleva a cabo en el inicio de la simulación, tras medir las pérdidas de propagación y condiciones de recepción entre los ED y el GW que les dé servicio, tal y como contempla la infraestructura de red LoRaWAN presente en [4]. Por otro lado, se considera una localización estática de los diferentes dispositivos de la red para facilitar la recolección y análisis de datos obtenidos de acuerdo a la localización de los nodos y las métricas establecidas. El estudio de datos bajo escenarios de movilidad puede subyacer del modelo estático, sin embargo, queda fuera del alcance del presente trabajo.

La implementación de cada uno de los dos escenarios realizada en el presente trabajo se encuentra disponible en el repositorio [37] bajo los nombres de *lora-open-area* y *lora-urban-area.* Ambos siguen una estructura y secuenciación similar, que hace uso de los diferentes modelos de red *LoraWAN* disponibles en [4] y [6] así como del modelo de energía del presente trabajo. La secuencia de actividad está formada por la configuración inicial de los parámetros de simulación, la creación de los edificios (escenario urbano), el establecimiento del modelo de localización de los dispositivos, la creación y configuración del canal de transmisión, la instalación del *stack* de comunicación en los EDs y GW, la asignación de los SF en los EDs en función de las condiciones de pérdidas de propagación de los dispositivos, la instalación de los dispositivos en el *layout* de

edificios generado (escenario urbano), el establecimiento de los *periodic senders* en los EDs, la agregación del modelo de energía en los EDs, y el disparo de la simulación y recogida de datos. En cada uno de los escenarios, un número configurable de sensores (EDs) llevan a cabo un envío periódico de paquetes de datos de 20Bytes hacia los receptores (GWs), de acuerdo a una periodo de envío ajustable, bajo un rango frecuencial de 868 MHz, un *Duty Cycle* de 1% y una potencia de transmisión 14 dBm que responde a las condiciones más favorables de envío dentro de las limitaciones de ETSI y que posee un valor asociado de consumo de corriente en modo TX de 43.5 mA según el modelo de interpolación de consumo de corriente implementado. Por su parte, el consumo de corriente en modo RX es de 112 mA, el de STANDBY de 1.4mA y el de SLEEP 1.8μA, tal y como se ha indicado en 5.3.3. Los escenarios contemplan dos topologías de red: topología de estrella constituida por un único receptor en el centro del área de simulación, y topología estrella de estrellas, constituida por tres receptores distribuidos de forma uniforme a lo largo del escenario para dar cobertura a la mayor cantidad de dispositivos desplegados en el mismo.

El escenario de área abierta consiste en el despliegue de una red de dispositivos en un área cuadrada de 20 Km de lado ($400\ Km^2$) de acuerdo a una distribución uniforme $P(x,y)\sim U(r)$ y el modelo de pérdidas de propagación en espacio libre descrito en 4.1.1. Los sensores poseen una altura de 1.5 m, coherente con la característica del terreno en escenarios agrícolas, y los receptores una altura de 15 m emulando una posición de terreno superior idónea para la recepción.

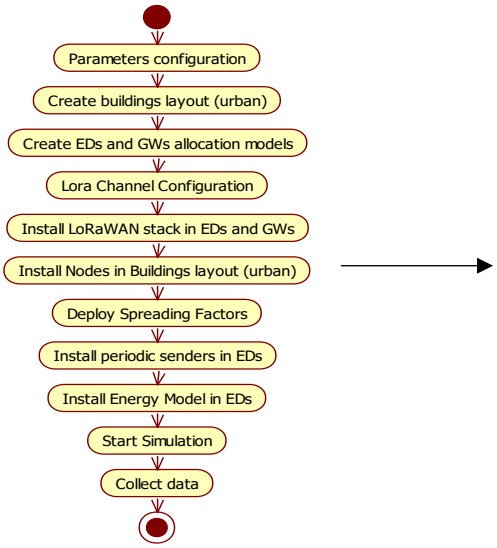


Figura 43 Secuencia de simulación de escenarios

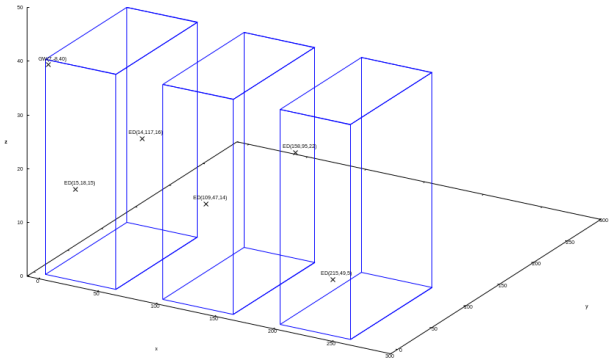


Figura 44. Ejemplo de generación e instalación de nodos en edificios. Datos capturados por la clase *LoraStatsHelper* y generados por GNUplot.

El escenario de área urbana consiste en el despliegue de una red de dispositivos en un área cuadrada de 4 Km de lado ($16\ Km^2$) en él se ubica una serie de bloques de edificios con unas dimensiones de 60 x 120 x 40 m, emulando las dimensiones presentes en una gran ciudad, de acuerdo a una distribución uniforme $P(x,y,z)\sim U(r)$, y el modelo híbrido de pérdidas de propagación descrito en 4.1.1. Los sensores, pueden ubicarse en el interior o en el exterior de los

edificios, disponiendo de un valor de altura aleatorio en el rango de [1.5m, 5m] si encuentran en el exterior, emulando así su posición en diferentes elementos del entorno urbano (semáforos, interior y fachadas de edificios, etc.), mientras que los receptores poseen una altura fija de 30 m, emulando una situación favorable de operación bajo este tipo de escenarios.

5.7 Métricas

Para llevar a cabo el análisis de los datos generados en los escenarios de simulación, así como verificar la validación del modelo energético es necesario establecer una serie de métricas que darán consistencia al modelo:

- Los sensores (ED) que operen bajo condiciones de transmisión con mayores pérdidas de propagación poseerán un mayor consumo energético. Cuanto mayor son las pérdidas por propagación, mayor será el SF asignado al dispositivo al inicio de la simulación, y consecuentemente mayor será el tiempo de propagación en el aire (TOA) de los paquetes enviados por los dispositivos. Ante un mayor tiempo TOA por paquete enviado, mayor es el tiempo en modo de operación TX, caracterizado por un mayor consumo de corriente en comparación con el resto de los modos de operación.

- Como consecuencia directa de la métrica anterior, el radio que caracteriza la operación bajo condiciones de mayor eficiencia energética, definido desde el punto de localización de los GWs hasta aquellos dispositivos que presentan un alto rendimiento energético en comparación con el resto de dispositivos, será menor en escenarios urbanos, debido a que las pérdidas por propagación que presenta son mayores que las que caracterizan el escenario agrícola.

- La modificación de la topología de los escenarios (estrella o estrella de estrellas) implicará un cambio en la distribución de SF en los dispositivos, lo que implicará una modificación del consumo de energía de los dispositivos que se vean afectados por una reasignación del SF al inicio de la simulación.

- La variación del patrón de envío de paquetes por parte de los EDs repercutirá en su consumo energético. Un periodo menor en el ciclo de envío de paquetes conllevará un mayor consumo energético de los EDs dada una mayor frecuencia de operación en modo TX.

- Dado que la simulación contempla únicamente el sentido ascendente de la red LoRaWAN, el tiempo en modo de operación RX será nulo, al igual que su consumo energético. Pese a que los dispositivos efectúan la apertura de ventanas de escucha tras la transmisión de un paquete, el modo de funcionamiento del dispositivo permanece en STANDBY hasta que la capa PHY detecta condiciones de recepción válida, situación que no se produce debido a que no se efectúan envíos de paquetes en sentido DL.

- En estado de no envío, el consumo energético en modo SLEEP deberá ser superior al de STANDBY dado que, a pesar de que el consumo de este último es de varios órdenes superior, éste sólo está presente en los momentos previos al envío de paquetes y a la apertura de ventanas de recepción, intervalo muy inferior al correspondiente del modo SLEEP bajo el cual opera mayoritariamente el dispositivo, compensando con ello la diferencia de consumo de corriente en ambos modos.

- El consumo de los dispositivos ED deberá ser mayor cuanto mayor sea el valor de SF bajo el que operan. Este hecho implica a su vez que en los escenarios donde haya visión directa entre dispositivos (LOS), cuanto mayor sea la distancia entre el GW y los ED que comunican con él, mayor será el valor del consumo medio de éstos últimos.

- En el cómputo del consumo global de los dispositivos deberá tenerse en cuenta la ausencia del modo de operación RX de los dispositivos en los escenarios de operación, lo que repercutirá en un valor menor de consumo con respecto a los *benchmarks* llevados a cabo en las referencias de 5.2, y consecuentemente una mayor estimación del tiempo de vida de la batería.

6.Despliegue y resultados

A continuación, se presentan los resultados de la simulación de los dos escenarios anteriormente descritos bajo el establecimiento de diferentes parámetros de configuración que permitirán realizar un estudio de las métricas anteriormente planteadas. Se han establecido bajos tiempos de simulación para rebajar la carga computacional debido al gran número de nodos implicados en la misma, hecho que no condiciona el estudio de tiempos de mayor magnitud dado que se pueden derivarse directamente. Por otro lado, en el análisis del presente apartado se utilizan dos únicas configuraciones de periodicidad de envío de paquetes. La primera, correspondiente a 5 transmisiones uniformemente distribuidas en un intervalo de 60 minutos, pretende analizar el consumo de dispositivos que poseen altos requerimientos de actualización y envío de datos, siendo coherente con las recomendaciones de la especificación LoRaWAN. La segunda, correspondiente a una única transmisión centrada en el mismo intervalo, se utiliza como ciclo base para evaluar otros parámetros de la simulación de acuerdo a las métricas establecidas en 5.7. Sigue siendo un escenario alineado con una aplicación real LoRaWAN, pero rebaja considerablemente la carga computacional de la simulación.

Para llevar a cabo la captura de datos y generación de imágenes a través de GNUPlot, se ha hecho uso de la clase *Lora-Stats-Helper,* implementada en el presente trabajo y disponible en [37].

6.1 Escenario agrícola (LOS).

La figura 45 muestra el despliegue de una red sensorial en una extensión de área abierta formada por un único GW en el centro del escenario que da servicio a 300 ED, en la cual se ha establecido una frecuencia de envío de datos desde los ED de 5 minutos, establecida de forma uniforme en un intervalo de 0 a 60 minutos.

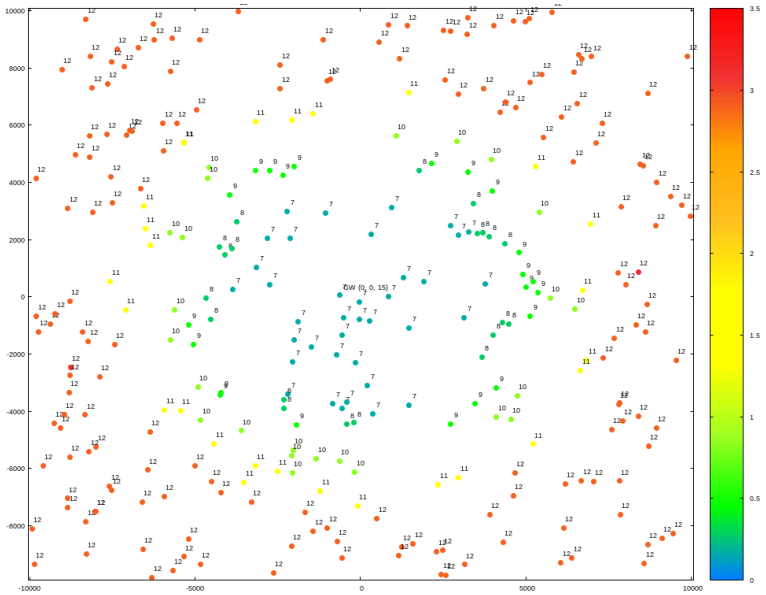


Figura 45. LOS. Estrella. 300 ED. Ciclo 5 minutos.

En ella puede identificarse el SF asignado a cada uno de los nodos de forma numérica y el grado de eficiencia energética a través de una marca de color asociada al gradiente de la derecha que varía en función del valor de energía, en julios, consumida por el dispositivo durante el tiempo de la simulación. Tal y como se observa en la misma, los dispositivos más cercanos al GW poseen un menor consumo energético como consecuencia de su operación bajo un SF menor. La tabla 11 muestra el consumo asociado a cada SF, tras el envío de 12 paquetes de 20 Bytes por parte de los dispositivos. Existe una diferencia de 2.7341 Julios entre los dispositivos que operan bajo un SF de 7 y un SF de 12, asociada a un mayor tiempo de TOA los dispositivos con alto valor de SF y consecuentemente a un mayor tiempo de operación TX.

SF	TX [J]	RX[J]	STANDBY [J]	SLEEP [J]	TOTAL [J]
7	0.129048	0.0000	0.0012432	0.0236768	0.153968
8	0.238319	0.0000	0.0012432	0.0232536	0.262816
9	0.437084	0.0000	0.0012432	0.0234878	0.461814
10	0.795057	0.0000	0.0012432	0.0220504	0.818351
11	1.59011	0.0000	0.0012432	0.0231818	1.61454
12	2.86379	0.0000	0.0012432	0.023097	2.88813

Tabla 11. Datos de consumo. LOS. Estrella. 300 ED. Ciclo 5 minutos.

La figura 46 muestra la simulación del escenario anterior en la cual se ha rebajado la frecuencia de envío de datos. En este caso los EDs envían un único paquete cada 60 minutos, cuyo disparo tiene lugar a los 1800s, centrado en el intervalo de simulación.

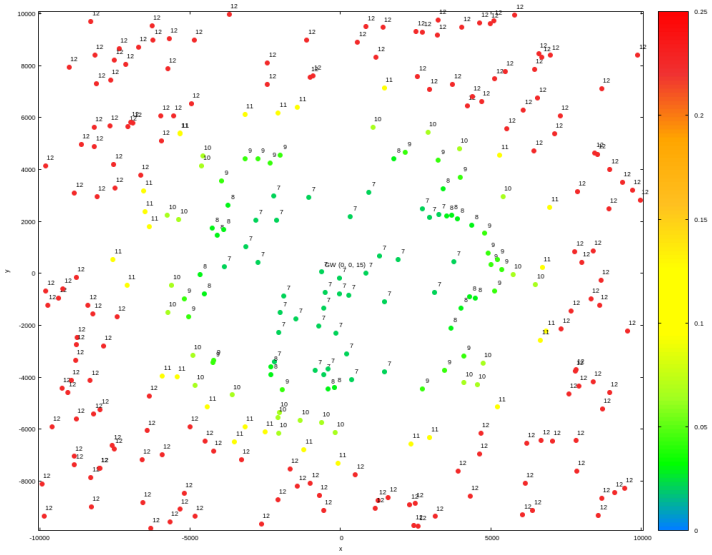


Figura 46. LOS. Estrella. 300 ED. Ciclo 60 minutos.

En este caso se observa una reducción del consumo de los dispositivos en cada uno de los SF de operación, asociada a una menor frecuencia de envío bajo un mismo intervalo de simulación. Por otro lado, cabe observar que el diferencial de consumo entre los dispositivos que presentan diferentes valores de SF es de 0.204J, un orden inferior a la situación anterior. Este hecho es debido a que a que el consumo en modo TX de los dispositivos tiene un peso menor en el

cómputo global de consumo con respecto a una situación de mayor frecuencia de envío, tal y como puede observarse en la tabla 12.

SF	TX [J]	RX[J]	STANDBY [J]	SLEEP [J]	TOTAL [J]
7	0.00828184	0.0000	0.0001036	0.0239755	0.0323609
8	0.0165637	0.0000	0.0001036	0.0239751	0.0406424
9	0.0298311	0.0000	0.0001036	0.0239746	0.0539093
10	0.0530697	0.0000	0.0001036	0.0239736	0.0771469
11	0.106139	0.0000	0.0001036	0.0239714	0.1302140
12	0.212279	0.0000	0.0001036	0.0239670	0.2363496

Tabla 12. Datos de consumo. LOS. Estrella. 300 ED. Ciclo 60 minutos.

Una vez comprobado cómo afecta la periodicidad del envío de paquetes bajo un mismo intervalo de simulación en el perfil de eficiencia energética del escenario, falta analizar la influencia de un cambio de topología en el mismo bajo unas mismas condiciones de envío de datos por la parte de la red sensorial. Para ello se ha aumentado la densidad de dispositivos desplegados en el escenario hasta un total de 1000 ED facilitando así la percepción en el cambio del perfil energético generado en la simulación. Por otro lado, en la simulación se ha efectuado un único disparo de envío bajo las mismas condiciones que el caso anterior, rebajando así la carga computacional en la simulación.

La figura 47 muestra una topología de red en estrella, donde se ha ubicado un único GW situado en el centro del escenario para dar servicio al conjunto de dispositivos desplegados en la simulación. En ella se observa como el diferencial de consumo energético entre diferentes SF coincide con el caso anterior dadas las mismas condiciones de operación

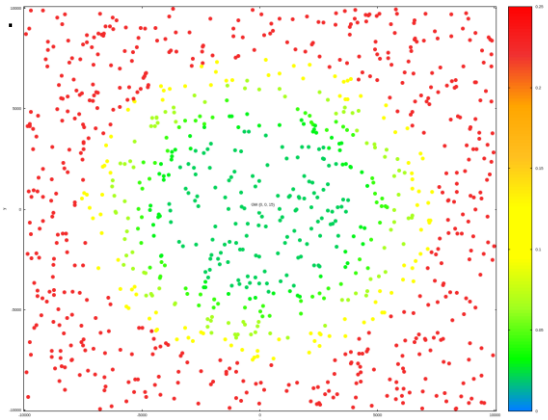


Figura 47. LOS. Estrella. 1000 ED. Ciclo 60 minutos.

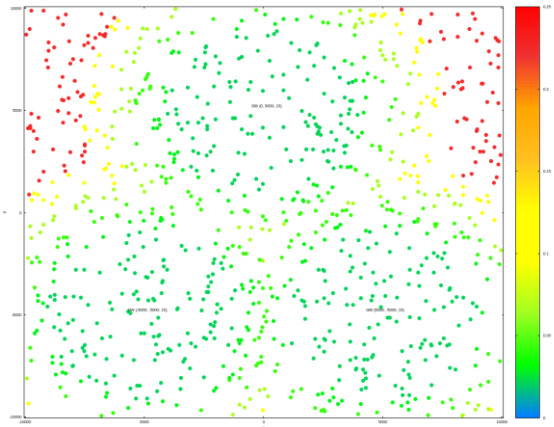


Figura 48. LOS. Estrella de estrellas. 1000 ED. Ciclo 60 minutos.

La figura 48 muestra el mismo escenario, pero bajo una topología de “estrella de estrellas”, constituida por tres GW uniformemente distribuidos emulando un despliegue LoRaWAN real. En ella se hace patente el cambio en el perfil de consumo energético del escenario dado un mayor área de despliegue de SF de bajo valor en comparación con el escenario que posee un único GW.

6.2 Escenario urbano (NLOS).

La simulación de escenarios urbanos pretende hacer patente la pérdida de rendimiento energético de los dispositivos que operan bajo una red LoRaWAN en situaciones desfavorables de pérdidas por propagación.

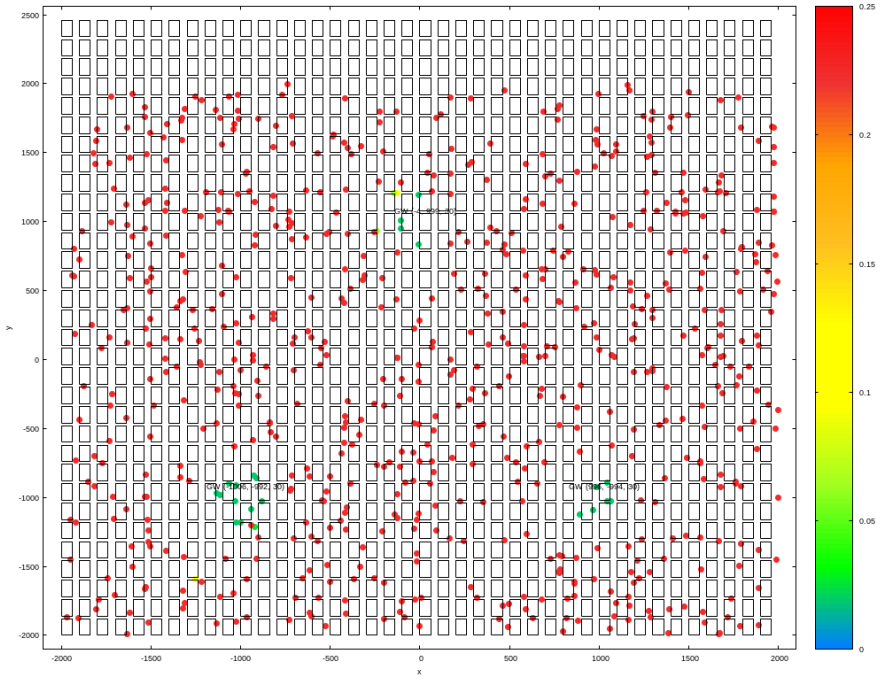


Figura 49. NLOS. Estrella de estrellas. 700 ED en el exterior. Ciclo 60 minutos.

La figura 49 muestra el despliegue de una red sensorial en una extensión de área urbana simulando el escenario *Smart City* y *Smart Home* descrito en 5.6, en la cual se han desplegado tres GW con una altura de 30m, emplazados de forma uniforme para dar servicio a 700 ED situados en el exterior de edificios (cuyas plantas pueden apreciarse en la imagen), que efectúan una única transmisión de 20 Bytes centrada en un intervalo de 60 minutos. En ella se observa cómo el radio de cobertura de los GW caracterizado por una alta eficiencia energética se reduce considerablemente en comparación con el escenario de área abierta. Pese a que la distancia entre GW es considerablemente menor que en el caso del escenario de área abierta, el número de dispositivos que operan con valores bajos de SF es mucho más reducido, conformando un radio de cobertura eficiente de unos 200 metros, en comparación los 5 km, que pueden apreciarse en la simulación de escenarios anteriores de visión directa. Esto pone de manifiesto el alto grado de influencia que tiene el modelo de propagación de los escenarios en la eficiencia energética de los dispositivos que operan en él.

La figura 50 muestra la simulación del mismo número de dispositivos, pero emplazados de forma aleatoria bajo una distribución uniforme en el interior de los edificios. En ella se observa cómo el perfil de consumo energético está menos definido en comparación con la situación anterior, dado que la altura a la cual se

emplaza a los ED está determinada por las dimensiones de los edificios, presentando un margen de 0 a 40 metros, en comparación con el caso anterior que presenta un margen de 1 a 5 metros, tal y como se indica en 5.6.

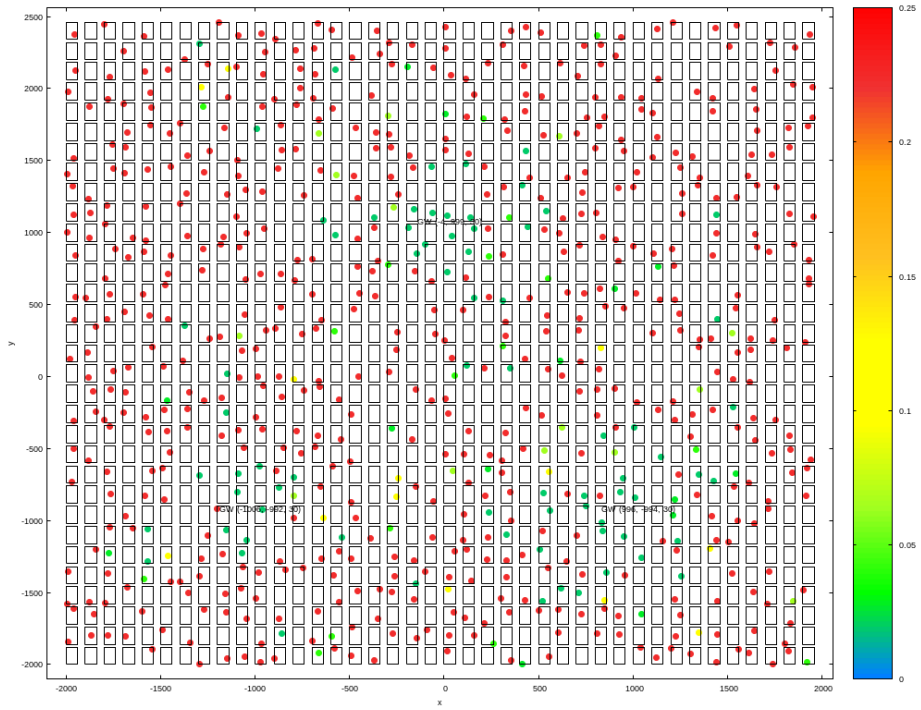


Figura 50. NLOS. Estrella de estrellas. 700 ED en el interior. Ciclo 60 minutos.

Por último, la figura 51 muestra la simulación bajo un mismo número de dispositivos, situados tanto en el interior como en el exterior (50% - 50 %), en la cual se ha elevado la altura de los GWs emplazados a 40m, emulando su localización en azoteas o tejados de los edificios. En este caso, el radio de eficiencia energética abarca prácticamente todo el escenario de simulación, logrando un radio de eficiencia energética comparable al establecido en escenarios de visión directa, poniendo de manifestó la importancia en el emplazamiento de los GW para garantizar un consumo eficiente de los dispositivos con los que intercambian datos.

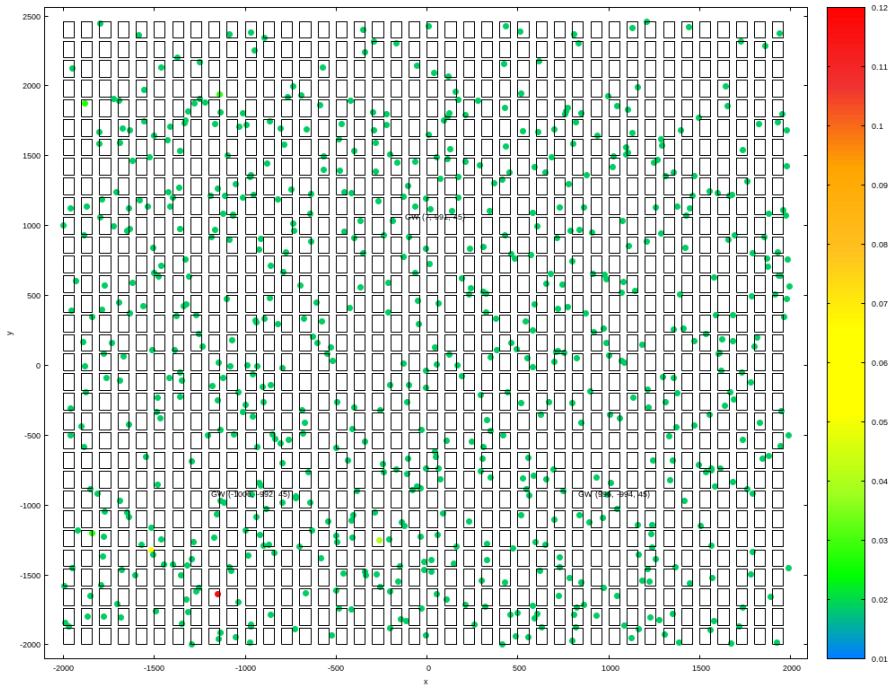


Figura 51. NLOS. Estrella de estrellas. 700 ED en interior. Ciclo 60 minutos. Altura de GW de 40 m.

6.3 Valoraciones

En los diferentes escenarios del apartado 6.3 se ha analizado cómo afecta la variación del entorno de propagación, la localización y altura de los nodos, la topología y la frecuencia de envío de datos, en el consumo medio de los dispositivos ED que operan en una red LoRaWAN.

En la tabla 11 se establece una comparación del radio de eficiencia energética en cada uno de los escenarios contemplados, considerando éste como aquel que abarca a los dispositivos que presentan el menor consumo (marca verde) en el área de simulación. Tal y como se observa en la misma, los escenarios con visión directa poseen un alcance más de un orden mayor que aquellos basados en áreas urbanas.

Escenario	Radio de eficiencia energética
LOS – Estrella	5000 m
LOS – Estrella de estrellas	> 7500 m
NLOS – Localización externa de ED, altura variable [1.5m, 5m]	200 m
NLOS – Localización de ED en el interior de edificios, altura variable [0m, 40m]	> 200 m

Tabla 13. Radio de eficiencia energética en los diferentes escenarios simulados.

El escenario LOS con un único GW central, presenta un radio de eficiencia energética de 5000 m, mientras que el mismo, siguiendo una topología de "estrella de estrellas," presenta un radio de 7500m. Esto pone en evidencia la importancia que tiene la densidad de receptores en las topologías LoRaWAN para garantizar un nivel eficiente de los dispositivos ED que operan en ellas. Por otro lado, en los escenarios NLOS puede observarse cómo se hace patente además la necesidad de definir adecuadamente la distancia entre GWs para ampliar de forma óptima el espectro de eficiencia energética de los dispositivos, y evitar zonas "rojas" de consumo de los dispositivos. A su vez, se hace evidente también la influencia que tiene el posicionamiento de los nodos en el nivel de consumo energético. De acuerdo a lo anterior, para mejorar el rendimiento energético de los dispositivos en redes LoRaWAN habrá que garantizar un volumen de densidad de receptores o GW, tal que asegure el solapamiento entre los radios de eficiencia energética de cada uno de los receptores. Establecer *grids* hexagonales como los utilizados en tecnologías celulares sería una opción a considerar dado su alto nivel de granularidad en el despliegue de estaciones base para evitar "zonas muertas" de cobertura.

En el escenario NLOS donde se establece un posicionamiento aleatorio de los nodos en el interior de los edificios, puede observarse la presencia de nodos aislados que operan con un nivel eficiente de consumo. Esto es debido a que presentan condiciones de localización (especialmente altura) más favorables que otros que operan en áreas cercanas con un nivel mayor de consumo.

En cuanto a la medida de eficiencia energética de los dispositivos, en las tablas 14 y 15 pueden observarse los valores de consumo diario y anual de los dispositivos para una frecuencia de envío de 5 y 60 minutos respectivamente en función del SF utilizado. Los valores de consumo capturados merecen una especial atención dado que en ellos se hace evidente las limitaciones de las condiciones de simulación del entorno utilizado.

SF	Consumo diario [mAh]	Consumo anual [mAh]
7	0.27702	101.1123
8	0.47306	172.6701
9	0.8312	303.4117
10	1.8018	657.6576
11	2.5428	928.1357
12	5.1985	1897.4817

Tabla 14. Relación entre SF y consumo de acuerdo a una frecuencia de envío de 5 minutos.

SF	Consumo diario [mAh]	Consumo anual [mAh]
7	0.0582	21.26
8	0.07308	26.6742
9	0.09702	35.4123
10	0.1388	50.6809
11	0.2344	85.5545
12	0.4253	155.2491

Tabla 15. Relación entre SF y consumo de acuerdo a una frecuencia de envío de 60 minutos.

En ellas puede observarse cómo el consumo anual de los dispositivos es inferior a lo contemplado en las medidas realizas en cada una de las referencias indicadas en 5.2, superando con ello incluso la predicción que se establece en la literatura de una duración de batería de 10-15 años para una frecuencia de envío comparable a las establecidas en la simulación. Esta variación es coherente, sin embargo, con las asunciones realizadas en el despliegue de los escenarios como consecuencia de las limitaciones que presenta la plataforma de simulación utilizada para realizar la integración del modelo de energía. En ella, no se ha contemplado la operación de los dispositivos en estado RX, donde la corriente de consumo asociada es varios órdenes mayores que la asociada a la del estado de reposo de los dispositivos. A su vez, los dispositivos operan bajo el estado SLEEP durante todo el intervalo inactivo de la simulación a excepción de los intervalos correspondientes a las ventanas de recepción donde operan bajo el estado de STANBY, asumiendo con ello unas condiciones de consumo eficiente, hecho que puede variar en función de la configuración de los dispositivos utilizada para llevar cabo las medidas de consumo, tal y como se contempla en [26].

Estas desviaciones, sin embargo, están asociadas a las limitaciones de la infraestructura disponible en [4] para simular escenarios reales en el módulo LoRaWAN que contemplen transferencia de datos en el enlace descendente (DL), así como interfaces para establecer diferentes configuraciones y parámetros en los modos de operación de los dispositivos. El seguimiento del estado de operación de los dispositivos, así como el cálculo del consumo energético en función del tiempo de operación en cada uno de ellos es competencia del módulo de energía del presente trabajo, el cual ha seguido el proceso de validación descrito en 5.5, sin embargo, la caracterización del *stack* de comunicación y su operación en los escenarios queda fuera del alcance del presente trabajo, asumiendo que su validación se lleva cabo en [4].

Considerando las limitaciones anteriores, habría que asumir, por tanto, un incremento notable del consumo de los dispositivos si se contempla la operación en RX en los escenarios simulados, donde los resultados obtenidos no superarían los estudios realizados con dispositivos bajo escenarios reales. Sin embargo, esto requiere de futuras simulaciones una vez que dichas limitaciones no se encuentren presentes en los escenarios a aplicar el modelo energético. Una vez que se encuentre la infraestructura en [4] para llevar a cabo simulaciones más cercanas a condiciones reales, donde no haya que asumir las limitaciones anteriores, el acoplo del modelo de energía es automático, dado que su caracterización es independiente de la lógica de operación de los dispositivos.

7. Conclusiones

El presente trabajo ha tenido como objetivo llevar a cabo la integración de un modelo de energía en la simulación de redes LoRaWAN bajo el entorno de simulación basado en eventos discretos NS – 3, tomando como punto de partida un *stack* de red LoRa ya implementado en un repositorio que hace uso de la plataforma. Para cumplir los objetivos contemplados en el alcance, se han establecido una serie de fases y paquetes de trabajo claramente definidos en la planificación inicial, así como una clara estructuración de su contenido.

Con el objetivo de contextualizar el problema a resolver, se ha realizado un análisis previo de las principales tecnologías LPWAN que han irrumpido en los últimos años con gran fuerza en el ámbito de *Internet of Things,* para así determinar el interés de la aplicación del modelo energético en LoRaWAN frente a otras redes de características similares. A su vez, se han descrito las partes más relevantes de la especificación LoRaWAN, pasando por su arquitectura, capa física y capa de enlace, como estudio previo al análisis de su modelo de simulación implementado en NS-3.

Completado el estudio teórico de la red, se ha realizado un breve análisis de la plataforma NS-3, donde se ha detallado la estructura por capas en la cual está basada su arquitectura, así como los términos claves ampliamente utilizados en los diferentes modelos simulados, para realizar posteriormente un análisis pormenorizado de los modelos de red y *stack* de LoRaWAN del módulo que ha sido utilizado como punto de partida del presente trabajo para realizar la integración del modelo de energía.

El proceso de integración del modelo de energía se ha dividido en una fase previa de diseño teórico de los elementos que forman parte del modelo, para posteriormente realizar el diseño y la arquitectura de software haciendo uso del *framework* de energía de NS-3 y tomando como referencia la implementación de modelos de energía implementados en otros módulos de red. Posteriormente se ha llevado a cabo la codificación del mismo siguiendo un protocolo de validación y se han creado varios escenarios de aplicación para llevar a cabo un análisis posterior, en el cual se han encontrado desviaciones con respecto a los estudios presentes en la literatura debido a las limitaciones del entorno de simulación de escenarios.

El trabajo ha tenido una alta carga de análisis en las fases iniciales que ha requerido de un tiempo superior al inicialmente contemplado, debido especialmente a la complejidad del entorno de simulación utilizado y al análisis y comprensión del modelo LoRaWAN ya implementado, en el cual se realiza la integración del modelo de energía. A su vez, la fase de codificación ha tenido una carga muy superior a la contemplada en el inicio del proyecto, debido principalmente a la no disponibilidad de varios modelos auxiliares para la creación de escenarios en la simulación, y a la necesidad de creación de mecanismos de captura de datos y generación de gráficas para realizar análisis posteriores a las simulaciones. Todo ello ha ocasionado desviaciones en la planificación inicial del trabajo, lo que ha requerido de una revisión continua y un

reporte periódico del estado de las diferentes actividades que han formado parte de los paquetes de trabajo en el que se ha desglosado el trabajo.

Con respecto a los objetivos contemplados en el alcance, cabe señalar que se han cumplido prácticamente en su totalidad, sin embargo, tal y como se indicado en las valoraciones de los resultados, los mecanismos de simulación de escenarios todavía presentan ciertas carencias que repercuten en la medida del consumo promedio de los dispositivos, lo que ha impedido realizar una comparación óptima con los estudios que pueden encontrarse en la literatura.

Las incidencias que han tenido lugar a lo largo del desarrollo del trabajo se han materializado en una serie de lecciones aprendidas que se tendrán en cuenta en la implementación de futuros trabajos para reducir la desviación entre los objetivos inicialmente establecidos y los conseguidos tras finalizar el proyecto. Entre ellas, por su impacto en el presente trabajo, cabe señalar la necesidad analizar con detalle la infraestructura bajo la cual se parte de base, así como las herramientas disponibles en la misma para dimensionar correctamente la carga de trabajo y definir objetivos coherentes con las limitaciones temporales establecidas. A su vez, en el establecimiento de objetivos, debe tenerse en cuenta estas limitaciones para localizar posteriormente las causas de las posibles desviaciones observables en el análisis de los resultados.

En cuanto al establecimiento de líneas futuras, cabe señalar la necesidad de la introducción de una serie de mejoras que no han sido contempladas en el alcance inicial del presente trabajo:

- Implementación de la infraestructura en el módulo LoRaWAN que permita la creación de escenarios con transmisiones tanto en sentido ascendente (UL) como descendente (DL) del enlace, estableciéndose mecanismos de control, así como métodos de espera de confirmación (ACK), y mecanismos adaptativos de tasa de bits ADR.
- Creación del modelo *Mac* de dispositivos de clase B, y C en la simulación *stack* de LoRaWAN, para llevar a cabo estudios de eficiencia energética entre las diferentes clases de dispositivos que contempla la especificación.
- Implementación de mecanismos de posicionamiento que permitan crear topologías de red más densas y uniformemente distribuidas en los escenarios simulados.
- Ampliar el estudio de eficiencia energética a otras bandas frecuencias y de acuerdo a diferentes valores de limitación del *Duty Cycle.*

8. Glosario

ACK: Acknowledge.
ADR: *Adaptive Data Rate*.
ASK: *Amplitude-Shift Keying.*
API: *Application Programming Interface.*
BPSK: *Binary Phase Shift Keying.*
BER: *Bit Error Rate.*
CR: *Code Rate*
DCE: *Direct Code Execution.*
DL: *Down Link.*
DR: *Data Rate*
DSS: *Direct Sequence Spread Spectrum*
ED: *End Device.*
ERP: *Effective Radiated Power.*
EWL: *External Wall Loss.*
FEC: *Forward Error Correction*
FSK: *Frequency-Shift Keying*.
FHSS: *Frequency Hopping Spread Spectrum.*
GTK: *GIMP Toolkit.*
GW: *Gateway.*
HTTP: *Hypertext Transfer Protocol.*
IOT: *Internet of Things*.
IP: *Internet Protocol.*
ISM: *Industrial, Scientific and Medical.*
IWL: *Internal Wall Loss.*
LPLAN: *Low Power Local Area Network.*
LPWAN: *Low Power Wide Area Network.*
LOS: *Line of Sight.*
NLOS: *Non Line of Sight.*
NS: *Network Server*.
POSIX: *Portable Operating System Interface.*
PSK: *Phase-Shift Keying.*
RPMA: *Random Phase Multiple Access.*
OTA: Over the Air.
SF: *Spread Factor.*
SHA: *Secure Hash Algorithm.*
SINR: *Signal to Interference plus Noise Ratio.*
SISO: *Single Input Singe Output.*
SLA: *Service Level Agreement.*
SSH: *Secure Shell.*
SS: *Spread Spectrum.*
UNB: *Ultra Narrow Band.*
UP: *Up Link.*
TDOA: *Time Difference of Arrival.*
WPAN: *Wireless Personal Area Network*
XML: *eXtensible Markup Language.*

9. Bibliografía

[1] D. Magrin, M. Centenaro and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario,". 2017. Disponible [2018]: http://ieeexplore.ieee.org/document/7996384/
[2] Git. 2018. https://git-scm.com/
[3] Mercurial. 2018. https://www.mercurial-scm.org
[4] GitHub.DvdMgr.LoRaWAN. 2018. https://github.com/DvdMgr/LoRaWAN
[5] Waf. 2018. https://waf.io/book/
[6] Nsnam. 2018. https://www.nsnam.org/
[7] Semtech. 2018. https://www.semtech.com/
[8] 3GPP, "Radio Frequency (RF) system scenarios," Tech. Rep. 36.942 V13.0.0, Jan. 2016. Disponible [2018]: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2592
[9] T. Petrić, M. Goessens, L. Nuaymi, A. Pelov, and L. Toutain, "Measurements, Performance and Analysis of LoRa FABIAN, a realworld implementation of LPWAN," 2016, working paper or preprint. Disponible [2018]: https://hal-institut-mines-telecom.archives-ouvertes.fr/hal-01331966
[10] 3GPP."Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT)," Tech. Rep. 45.820 V13.1.0, Nov. 2015. Disponible [2018]: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2719
[11] R. Fraile, J. F. Monserrat, J. Gozálvez, and N. Cardona, "Mobile radio bidimensional large-scale fading modelling with site-to-site cross-correlation," European transactions on telecommunications, vol. 19, no. 1, pp. 101–106, 2008.
[12] S. S. Szyszkowicz, H. Yanikomeroglu, and J. S. Thompson, "On the Feasibility of Wireless Shadowing Correlation Models," IEEE Transactions on Vehicular Technology, vol. 59, no. 9, Nov. 2010. Disponible [2018]: https://ieeexplore.ieee.org/document/5590312/
[13] C. Goursaud and J.-M. Gorce, "Dedicated networks for IoT: PHY/MAC state of the art and challenges," EAI endorsed transactions on Internet of Things, 2015. Disponible [2018]: https://hal.archives-ouvertes.fr/hal-01231221
[14] J. C. Ikuno, M. Wrulich, and M. Rupp, "System level simulation of LTE networks," in Proc. IEEE Vehicular Technology Conference (VTC), May 2010, pp. 1–5. Disponible [2018]: https://ieeexplore.ieee.org/document/5494007/
[15] Mdpi.A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. 2018. http://www.mdpi.com/1424-8220/16/9/1466
[16] Anupriya K., Jerin Yomas, Jubin Sebastian E., "A review on IoT protocols for long distance and low power". Vol.5, No.6, December 2015. Disponible [2018]: https://pdfs.semanticscholar.org/f5b8/d7c2a4fd9e3480b6d10391bb29096af23034.pdf

[17] Johanna Nordlöf, Petter Lagusoon, “A Study of Low-Power Wide-Area Networks and an In-Depth Study of the LoRaWAN Standard”. 2017. Disponible [2018]:
https://kth.diva-portal.org/smash/get/diva2:1141920/FULLTEXT01.pdf
[18] Jean-Paul Bardyn, Thierry Melly, Olivier Seller, “IoT: The Era of LPWAN is starting now”.
https://ieeexplore.ieee.org/document/7598235/
[19] Koustabh Dolui, Soumya Kanti Datta,“ Comparison of Edge Computing Implementations: Fog Computing, Cloudlet and Mobile Edge Computing”. 2017. Disponible [2018]:
https://ieeexplore.ieee.org/document/8016213/
[20] Radar. O’Reilly. 3 Topologies driving IoT networking standards 2018.
http://radar.oreilly.com/2014/04/3-topologies-driving-IoT-networking-standards.html
[21] U. Raza, P. Kulkarni, and M. Sooriyabandara, “Low power wide area networks:An overview”, IEEE Communications Surveys Tutorials, vol. PP, no. 99, pp. 1–1, 2017. Disponible [2018]:
https://ieeexplore.ieee.org/document/7815384/
[22] Mattia Rizzi, Paolo Ferrari, Alessandra Flammini, Emiliano Sisinni, “Evaluation of the IoT LoRaWAN Solution for Distributed Measurement Applications”, vol. 66, no. 12, Dec. 2010. Disponible [2018]:
https://ieeexplore.ieee.org/document/8036410/
[23] Dmitry Bankov, Evgeny Khorov, Andrey Lyakhov, “On the limits of LoraWAN Channel Access”, 2016. Disponible [2018]:
https://ieeexplore.ieee.org/document/7810745/
[24] LoRaWAN 1.1 Specification. 2018. http://net868.ru/assets/pdf/LoRaWAN-v1.1.pdf
[25] 3glteinfo.LoRa Tutorial. 2018. http://www.3glteinfo.com/lora
[26] Phui San Cheong, Johan Bergs, Chris Hawinkel, Jeroen Famaey ”Comparison of LoRaWAN Classes and their Power Consumption”, 2017. Disponible [2018]:
https://ieeexplore.ieee.org/document/8240313/
[27] Wireless Solutions.Im880B-L. 2018. https://wireless-solutions.de/products/radiomodules/im880b-l.html
[28] Ncbi.Modeling the Energy Performance of LoRaWAN.2018.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5677147/
[29] José Daniel Rodríguez Nunca, “Dispositivo LoRa de comunicación a largo alcance y bajo consumo energético para aplicaciones del ámbito del desarrollo”, 2016. Disponible [2018]:
http://oa.upm.es/44890/1/TFM_JOSE_DANIEL_RODRIGUEZ_MUNCA.pdf
[30] Dorji. DRF1278F. 2018. http://www.dorji.com/docs/data/DRF1278F.pdf
[31] Internetofthingsagenda. IoT battery outlook: Types of batteries for IoT devices. 2018.
https://internetofthingsagenda.techtarget.com/feature/IoT-battery-outlook-Types-of-batteries-for-IoT-devices
[32] Bateriasonline. El Internet de las cosas (IoT) y sus limitaciones por las baterías. 2018.https://bateriasonline.com/es/blog/el-internet-de-las-cosas-iot-y-sus-limitaciones-por-las-baterias-n5
[33] Iot-architect. IoT Use Case-Battery Powered Device. 2018. https://www.iot-architect.de/iot-use-case-battery-powered-device

[34] Richtek. Designing Applications with Li-ion Batteries. 2018. http://www.richtek.com/battery-management/en/designing-liion.html
[35] GNUPlot. 2018. http://www.gnuplot.info/
[36] Itu.P.1411. 2018. https://www.itu.int/rec/R-REC-P.1411/es
[37] GitLab.Gdobato.LoRaWAN-EnergyModel . 2018. https://gitlab.com/gdobato/LoRaWAN-EnergyModel
[38] Itu.P.1238. 2018. https://www.itu.int/rec/R-REC-P.1238/es
[39] P.Schneider, F. Lambrecht, A. Baier, "Enhancement of the Okumura-Hata propagation model using detailed morphological and building data", vol. 1, 1996. Disponible [2018]: https://ieeexplore.ieee.org/document/567508/
[40] Nsnam.HybridBuildingsPropagationLossModel. 2018. https://www.nsnam.org/docs/models/html/buildings-design.html
[41] A.M.D Turkmani, J.D.Parsons, D.G.Lewis "Measurement of building penetration loss on radio signals at 441, 900 and 1400MHz", vol. 58, 1988. Disponible [2018]: https://ieeexplore.ieee.org/document/5261594/
[42] Nsnam.ITUR1411. 2018. https://www.nsnam.org/docs/release/3.17/doxygen/classns3_1_1_itu_r1411_los_propagation_loss_model.html#details
[43]D.Magrin, L. Vangelista, M. Centenario "Network level performances of a LoRa system". 2017. Disponible [2018]: http://tesi.cab.unipd.it/53740/1/dissertation.pdf
[44] Nsnam.LogDistancePropagationLossModel. 2018. https://www.nsnam.org/docs/models/html/propagation.html#logdistancepropagationlossmodel
[45] Nsnam. EnergyModels. 2018. https://www.nsnam.org/doxygen/group__energy.html
[46] Nsnam. WifiRadioEnergyModel. 2018. https://www.nsnam.org/doxygen/wifi-radio-energy-model_8cc.html
[47] SiliconLightWorks.LiIonVoltage. 2018. https://siliconlightworks.com/li-ion-voltage
[48]Medium.Iotforall.yourprimerforloralorawan. 2018. https://medium.com/iotforall/your-primer-for-lora-lorawan-33f1e0eb4215
[49]Telecoms.bouyguestelecomtolaunchloranetworkdedicatedtoiot. 2018. http://telecoms.com/413422/bouygues-telecom-to-launch-lora-network-dedicated-to-iot/
[50]Sghoslya.lora. 2018. http://www.sghoslya.com/p/lora_6.html
[51]Hackmd.S1kg6Ymo. 2018. https://hackmd.io/s/S1kg6Ymo-