UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

ANÁLISIS E IMPLEMENTACIÓN DE REDES INALÁMBRICAS DE SENSORES DETERMINISTAS ROBUSTAS

José Vera Pérez

Tutor: Víctor Miguel Sempere Paya

Cotutor Empresa: David Todolí Ferrandis

Trabajo Fin de Máster presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Máster en Ingeniería de Telecomunicación

Curso 2016-17

Valencia, 31 de agosto de 2017

Agradecimientos:

A mi familia por todo el apoyo y confianza depositados en mi durante todos estos años, no estaría donde estoy de no ser por sus enseñanzas.

A mis directores de proyecto Víctor y David por su dedicación y consejo, y por brindarme la oportunidad de trabajar junto a este gran grupo de investigación.

A mi pareja y amiga, por estar siempre a mi lado y porque sin ella no habría podido afrontar muchas situaciones difíciles. Gracias

Resumen

Las redes de sensores inalámbricas (WSN) ofrecen un gran potencial de futuro, puesto que permiten cubrir un elevado número de puntos de sensorización en una superficie a un bajo coste y sin la utilización de cableado. Sin embargo, esta tecnología aún ofrece ciertos inconvenientes que están retrasando su penetración en el mercado: la complejidad para su despliegue y mantenimiento, y su baja fiabilidad.

Objetivos claros en el diseño de este tipo de redes abordan el bajo consumo de energía, o la robustez frente a los cambios de condiciones de la red. Adoptar protocolos de acceso al medio deterministas, como 802.15.4e TSCH, abren la posibilidad al despliegue de WSN preparadas para funcionar en los entornos más exigentes.

La temática del TFM trata del análisis de las tecnologías disponibles para WSN deterministas, la implementación y testeo de mecanismos que permitan construir una topología más robusta basándose en el nivel de batería de los nodos o la calidad de los enlaces, además de definir una planificación de acceso al medio de manera eficiente, con la finalidad de desplegar una red WSN con robusta y fiable.

Resum

Les xarxes de sensors sense fil (WSN) ofereixen un gran potencial de futur, ja que permeten cobrir un elevat nombre de punts de sensorització en una superfície a un baix cost i sense la utilització de cablejat. No obstant això, esta tecnologia encara ofereix certs inconvenients que estan retardant la seua penetració en el mercat: la complexitat per al seu desplegament i manteniment, i la seua baixa fiabilitat.

Objectius clars en el disseny d'este tipus de xarxes aborden el baix consum d'energia, o la robustesa enfront dels canvis de condicions de la xarxa. Adoptar protocols d'accés al mig deterministes, com 802.15.4e TSCH, obrin la possibilitat al desplegament de WSN preparades per a funcionar en els entorns més exigents.

La temàtica del TFM tracta de l'anàlisi de les tecnologies disponibles per a WSN deterministes, la implementació i testeig de mecanismes que permeten construir una topologia més robusta basant-se en el nivell de bateria dels nodes o la qualitat dels enllaços, a més de definir una planificació d'accés al mig de manera eficient, amb la finalitat de desplegar una xarxa WSN amb robusta i fiable.

Abstract

Wireless sensor networks (WSN) offer great potential for the future, since they allow to cover a large number of sensor on a surface on a low cost and without using wires. However, this technology still offers certain drawbacks that are delaying its penetration in the market: the complexity for its deployment and maintenance, and its low reliability.

Main objectives in the design of this type of networks are the low energy consumption, or robustness to the face up the changes in network conditions. Adopting deterministic medium access protocols, such as 802.15.4e TSCH, open the possibility to the deployment of WSNs which are prepared to operate in the most demanding environments.

The subject of this TFM is the analysis of the technologies available for deterministic WSNs, the implementation and testing of mechanisms allowing the construction of a topology that is more robust based on the battery level of the nodes or the quality of the links. Also, by defining a schedule for access to shared wireless medium efficiently, in order to deploy a robust and reliable WSN network.

Índice

Capítulo 1. Introducción, objetivos y estructura de la memoria

1.1 Introducción

Las comunicaciones inalámbricas son una de las mayores contribuciones a la tecnología en la historia de la humanidad. Este tipo de comunicaciones permiten transmitir información sin la necesidad de ningún medio cableado ni conductor, por lo que aporta mayor flexibilidad y sencillez a la hora de implantar un sistema de comunicaciones en cualquier tipo de escenario. Además, existen distintos sistemas que permiten cubrir diferentes distancias de transmisión, desde comunicaciones para cortas distancias como la que utiliza la tecnología de RFID, hasta intercambios de información intercontinental a través de comunicaciones por satélite.

Durante las últimas décadas nuestra sociedad ha experimentado grandes cambios gracias a importantes avances tecnológicos, como la aparición de la telefonía móvil e internet, y de nuevas formas de recoger, compartir, interpretar y representar la información. Todos estos avances han repercutido de forma contundente en nuestra calidad de vida: más seguridad, comodidad, comunicación y eficiencia en la interacción con nuestro entorno. Pero aún existe un enorme potencial de mejora, que generará una revolución en la forma de gobernar e interactuar con nuestras ciudades, en la eficiencia y calidad de los procesos de manufactura, o en la precisión y calidad de nuestros servicios de salud. Estos nuevos sistemas que están apareciendo son capaces de generar grandes cantidades de datos de interés que pueden ser utilizados en diferentes sectores. Estos sistemas irán acompañados de una necesidad de poder interactuar con el "mundo físico", utilizando para ello diferentes tipos de sensores y/o actuadores que ofrezcan una conectividad inteligente y transparente con estos sistemas.

Es aquí donde entra en juego el concepto del "Internet de las Cosas" (IoT de sus siglas en inglés), que surge para dotar de conectividad a Internet a cualquier tipo de objeto, permitiendo así interactuar con el resto de dispositivos que lo rodean de forma directa (conocido como comunicaciones M2M – Machine to Machine –). Dentro de las tecnologías habilitadoras de este tipo de aplicaciones encontramos las redes de sensores inalámbricas (WSN, de sus siglas en inglés), que suponen un pilar fundamental para el desarrollo y asentamiento del Internet de las Cosas.

Sin embargo, las WSN, especialmente las de tipo mallado, suponen todo un reto tecnológico debido a la complejidad que supone la interacción de un gran número de dispositivos en sistemas de comunicaciones sin infraestructura (comunicaciones ad-hoc), estableciendo conexiones en tiempo real, gestionando el tráfico, prioridades, calidad de servicio; y limitados a la vez por los recursos reducidos de este tipo de dispositivos, principalmente en cuanto a procesamiento, memoria y energía.

Pese a los grandes avances en WSN conseguidos por la comunidad científica, esta tecnología sigue ofreciendo importantes inconvenientes que provocan un retraso en su penetración en el mercado, principalmente en cuanto a fiabilidad y robustez de las comunicaciones: alta perdida

de paquetes, interferencias, altas latencias, complejidad de los protocolos de enrutamiento para soportar redes malladas, consumo de batería, incapacidad de ofrecer una garantía de servicio, etc.

En los últimos años han aparecido diversas propuestas y protocolos, como el estándar IEEE 802.15.4e, que incorpora un modo de acceso al medio llamado TSCH, y también protocolos de encaminamiento dinámico como RPL, que permite construir una topología dentro de una WSN que esté preparada para funcionar en los entornos industriales con requisitos de robustez, garantizando el cumplimiento de unos parámetros de calidad de servicio inalcanzables hasta ahora.

Aun así, estos estándares ofrecen sólo las bases sobre las que trabajar y desarrollar soluciones completas, facilitando los procesos y mecanismos que permiten construir las diferentes topologías de red, pero estando sujetos a posibles mejoras. En los últimos años, la comunidad científica se ha apoyado en estos protocolos y se están consolidando como estándares de facto para la implementación de las WSN, y presentan diferentes soluciones que permiten mejorar los mecanismos de acceso al medio mediante planificaciones más optimizadas o métodos que permiten escoger de manera más eficiente las rutas que se utilizarán por los dispositivos que formen la red.

En este proyecto final de master se pretende continuar este proceso de innovación y mejora de las comunicaciones mediante WSN, con el fin de mitigar las deficiencias que presentan estos sistemas para así conseguir mejorar su fiabilidad, robustez y en general su eficiencia para trabajar en escenarios industriales. Para ellos se realizarán diferentes implementaciones que se centrarán en distintos aspectos de mejora que pueden tener las WSN, como la parte de acceso al medio, el proceso de selección de rutas que utilizarán los diferentes nodos para encaminar la información o también optimizar los procesos de construcción de las topologías y del ciclo de trabajo para conseguir una mejora en la utilización de la energía, permitiendo que los nodos ahorren sus baterías y permitiendo que se extienda la vida útil de la red.

Este trabajo se ha realizado formando parte del grupo de investigación de Comunicaciones Avanzadas del Instituto Tecnológico de Informática (ITI), en el que colabora el Dr. Víctor M. Sempere Payá de la Universidad Politécnica de Valencia, tutor también del presente proyecto. Este grupo de investigación está formado por 4 investigadores y 2 profesores, y sus líneas de investigación principales están vinculadas a las redes de sensores inalámbricas y su integración en la industria 4.0: fiabilidad, robustez, seguridad, integración semántica transparente, arquitecturas IIoT y servicios de autogestión y soporte al despliegue. Estas actividades se desarrollan en el marco de diversos proyectos de investigación con financiación regional, nacional y europea, entre los que cabe destacar:

- Proyecto H2020 DEWI (Dependable Embedded Wireless Infraestructure).
- Proyecto H2020 BEinCPPS (Business Experiments in Cyber Physical Production Systems).
- Proyecto H2020 SCOTT (Secure COnnected Trustable Things).

1.2 Motivación

Aunque la utilización de protocolos de acceso al medio deterministas como IEEE 802.15.4e (TSCH) junto a RPL supone enormes ventajas para la utilización de las WSN, estas únicamente cubren algunos aspectos de la implementación de una red estable. Con la finalidad de conseguir una red WSN robusta y fiable hay que complementar estas tecnologías con optimizaciones de los protocolos de encaminamiento, algoritmos de planificación que permitan optimizar la asignación de recursos de capa MAC de manera que puedan hace frente a cambios dinámicos en la red, y en general, aquellos mecanismos que permitan que las WSN aporten mayor fiabilidad y robustez para favorecer su entrada en la industria y en el mercado.

1.3 Objetivos del proyecto

Los objetivos de este proyecto son:

- **Comprender los fundamentos teóricos de protocolos relacionados con las WSN**, entre los que se encuentran algunos más destacados como el estándar IEEE 802.15.4, protocolo de encaminamiento RPL o el nivel de adaptación a IPv6 como es 6LoWPAN.
- **Comprender y utilizar el sistema operativo ContikiOS** que nos permitirá integrar diferentes pilas de protocolos en dispositivos embebidos para poder construir redes WSN, tanto en despliegues con dispositivos reales como realizando simuladores en una plataforma software que viene incluida en el sistema operativo.
- **Analizar la capacidad de las funciones objetivo del protocolo de encaminamiento RPL** para poder construir redes que sean eficientes energéticamente, consiguiendo alargar la vida útil de los dispositivos que forman la WSN.
- **Desarrollar un mecanismo que permita obtener la calidad de los enlaces** que forman la WSN, lo que nos permitirá conocer mejor las localizaciones donde poder obtener mejor calidad, además de mejorar el rendimiento de la propia WSN a la hora de construir los enlaces entre los dispositivos.
- **Evaluar el comportamiento de diferentes planificaciones a nivel de capa MAC**.

1.4 Estructura de la memoria

En este apartado se explicará de forma breve como está estructurada esta memoria, de forma que al lector le sea sencillo ubicar las distintas partes que en ella se detallan.

En el Capítulo 2 se realiza un extenso análisis sobre el estado del arte actual en cuanto a protocolos de comunicaciones inalámbricos, sistemas comúnmente utilizados y tecnologías a emplear en el ámbito de las redes de sensores inalámbricas. Apoyándonos en este análisis, se justificará la selección de tecnologías utilizadas para el desarrollo de este trabajo.

En el Capítulo 3 se presenta la metodología que se ha seguido durante todo el desarrollo del proyecto, tanto a nivel individual como en conjunto con el grupo de trabajo CAINE del Instituto Tecnológico de Informática (ITI).

En el Capítulo 4 se detallarán las modificaciones realizadas en las distintas implementaciones de los protocolos, mecanismos utilizados para la realización de las pruebas, así como una evaluación de los resultados obtenidos en cada una de las áreas en las que se ha trabajado.

En el Capítulo 5 se desarrollarán las conclusiones finales a las que se ha llegado tras la realización del proyecto, además de incluir algunas propuestas pensadas para realizar en trabajos posteriores.

Finalmente se incluye un apartado de recopilación bibliográfica, donde se listan algunos de los documentos y referencias utilizados durante el desarrollo del proyecto. Esta bibliografía está correctamente referenciada a lo largo de toda la memoria. Además, se incluye un índice de tablas y figuras incluidas en el proyecto.

Capítulo 2. Estado del arte y selección de tecnologías

Las redes de sensores inalámbricas (Wireless Sensor Networks – WSN) han evolucionado en los últimos años hacia redes más seguras, fiables y robustas, lo que ha hecho que se convierta en una de las tecnologías más prometedoras del futuro y que sobre todo pueden resultar de gran interés en aplicaciones industriales relacionadas con la Automatización y el Control de procesos, ya que ofrecen grandes beneficios frente a las actuales redes cableadas, como el ahorro en costes de despliegue, flexibilidad en cuanto a la movilidad de los dispositivos sensores, reducción de la complejidad de la instalación, sencillez a la hora de expandir la red con nuevos dispositivos, siendo estas solo algunas de sus ventajas [2].

En un marco general, una WSN consiste en un gran número de pequeños sensores de bajo coste que se distribuyen en un área en concreto con el fin de recolectar datos de interés. Estos nodos disponen de diferentes sensores que permiten monitorear las condiciones físicas o ambientales, o también pueden disponer de actuadores que le permitan interactuar con el entorno físico a partir de la información recibida.

Este tipo de redes están pensadas para que puedan funcionar de manera ininterrumpida durante largos periodos de tiempo por lo que una de las consideraciones tecnológicas más importantes de estos dispositivos son sus capacidades de utilizar una cantidad mínima de energía y de poder gestionarla de manera eficiente. Estos dispositivos permiten crear redes que no necesitan de una estructura definida, denominadas redes ad-hoc, por lo que aporta mucha flexibilidad a la hora de definir distintos tipos de aplicaciones. Además, tienen la capacidad de reorganizarse automáticamente, siendo capaces de encaminar la información a través de diferentes caminos hasta llegar a un sistema central utilizando el medio inalámbrico. Las grandes posibilidades que ofrece disponer de grandes cantidades de información en tiempo real a un bajo coste comparado con las redes cableadas, permiten utilizar estas redes en un amplio abanico de aplicaciones [3] como:

- Monitorización de escenarios industriales.
- Monitorización de parámetros ambientales.
- Implantación de Smart Cities.
- Es un pilar fundamental para el Internet de las Cosas.
- Sistemas de alertas en caso de desastres naturales

Este capítulo se dividirá en tres partes en las que se seleccionarán las tecnologías y protocolos utilizados para el desarrollo de este proyecto. En primer lugar, se abordará la parte de los protocolos existentes en la actualidad para WSN con capacidades de trabajo en entornos industriales. A continuación, se compararán algunas de las plataformas de programación de sistemas embebidos destinados a las redes de sensores inalámbricas y finalmente algunas de las plataformas hardware que pueden utilizarse como nodos y Gateway a la hora implementar la red de sensores inalámbrica.

2.1 Selección de la pila de protocolos

En este apartado se pretende mostrar una visión global del conjunto de protocolos utilizados durante el desarrollo de este trabajo final de master, justificando la selección de cada uno de ellos para nuestro propósito. Tenemos que tener en cuenta que se trata de la implementación de una red de sensores inalámbrica destinada al ámbito industrial, por lo que protocolos deberán ser capaces de aportar unos requisitos mínimos de calidad en cuanto a robustez y fiabilidad de las comunicaciones. Las WSN pueden desplegarse utilizando diferentes tecnologías como el Bluetooth Low Energy [4] [5] [6] [7], aunque el protocolo que se está convirtiendo en estándar de facto a nivel de capa física es el protocolo 802.15.4, que es en el que nos centraremos y desarrollaremos toda la problemática a partir de él.

2.1.1 Estándar IEEE 802.15.4

El objetivo del grupo de trabajo IEEE P802.15 era definir un estándar que permitiera establecer comunicaciones de bajo coste utilizando dispositivos de bajo consumo. El estándar inicial, IEEE, Std 802.15.4-2003, definía únicamente dos opciones de capa física (PHYs), operando en diferentes bandas de frecuencia empleando un control de acceso al medio (MAC) simple y efectivo. En el año 2006 se realizó la primera revisión del estándar, añadiendo dos opciones más a la capa física. Esta revisión era retro compatible con su primera definición, pero añadía diferentes tramas de capa MAC, así como diferentes mejoras a dicha capa. Algunas de estas mejoras eran:

- Soporte para una base de tiempos compartida con un mecanismo de marcado temporal de los datos.
- Soporte para planificación de balizas o 'beacons'.
- Sincronización a través de mensajes broadcast en redes de área personal (PAN) con el modo de beacons habilitado.
- Mejoras a nivel de seguridad en capa MAC.

En 2011, el estándar se volvió a revisar para añadir las tres correcciones que se realizaron después de la revisión de 2006, añadiendo cuatro opciones más a la capa física y algunas capacidades a nivel de MAC. La revisión actual del estándar es la aprobada en el año 2015, IEEE Std 802.15.4-2015 [8], que recoge todas las correcciones realizadas desde el año 2011, incluyendo la enmienda IEEE 802.15.4e-2012 donde se incluyen diferentes modos de operación como TSCH, que será de gran interés para este proyecto.

El objetivo de este estándar, tal como está definido en el documento, es "definir las especificaciones de la capa física (PHY) y las subcapas de control de acceso al medio (MAC) para establecer una conexión inalámbrica de baja tasa de datos entre dispositivos fijos, portátiles y móviles con requerimientos de consumos de batería muy limitados. Además, el estándar proporciona diferentes modos de operación. La capa física está definida para dispositivos que operan en varias bandas de frecuencia que no requieren licencia, en una variedad de regiones geográficas".

El estándar está orientado al establecimiento de comunicación en redes de área personal de baja tasa de datos (LR-WPAN), este tipo de redes permite establecer comunicaciones de bajo coste que permiten diseñar aplicaciones con conectividad inalámbrica que presenten restricciones en cuanto a la energía que puedan consumir y una tasa de datos moderada. Los principales objetivos de una LR-WPAN son facilitar la instalación, proporcionar una transferencia de datos fiable, de muy bajo coste y una vida de las baterías razonable para este tipo de dispositivos embebidos. En este tipo de redes pueden participar principalmente dos tipos de dispositivos: aquellos que tenga una implementación completa de sus funciones, denominados full-function device (FFD), y otros cuya configuración limite sus capacidades, denominados reduced-function device (RFD). Los FFD implementan todas las características que puedan tener estos dispositivos, permitiendo así que ejerzan el papel de

coordinador de la red y siendo capaces de actuar como nodos intermedios para retransmitir la información de sus vecinos hacia su destino. Por otro lado, los RFD tienen limitadas sus funciones, permitiendo así optimizar su ciclo de trabajo con un número de tareas reducidas que permitan alargar la vida de sus baterías. Estos últimos nodos suelen tener capacidades sensoras o actuadoras y se basan en gestionar esta información para retransmitirla hacia un destino, que será el encargado de procesar dichos datos, eliminando la complejidad que requeriría hacerlo localmente.

Dependiendo de los requisitos que tenga la aplicación que se vaya a implantar, una red que utilice IEEE 802.15.4 puede operar bajo dos topologías: la topología en estrella o la topología peer-to-peer, en la que la comunicación se lleva a cabo entre un par de nodos dentro de su rango de cobertura. Ambas topologías se muestran en la Figura 1. En la topología en estrella la comunicación se establece entre cada uno de los diferentes dispositivos y un único controlador central, que será el coordinador de la red. Todos los dispositivos que forman parte de la red tienen una dirección única por la que pueden ser identificados para establecer la comunicación entre un origen y un destino, en este caso, los dispositivos dispondrán de dos tipos de direcciones, una corta y otra extendida que podrán ir incluidas en los campos correspondientes del paquete.

En la topología peer-to-peer también debe de haber un coordinador de la red de área personal, que al igual que en la topología en estrella se encargará de coordinar a todos los dispositivos que formen parte de la red. La diferencia de esta topología está en que los dispositivos se pueden comunicar con cualquier otro dispositivo que esté dentro de su rango de cobertura, sin necesidad de que sea el coordinador. De esta forma se podrían establecer comunicaciones que requieran de más de un salto para llegar hasta el destino, permitiendo así crear topologías de red más complejas como la topología mallada o topologías jerarquizadas. De esta forma se podrían cubrir mayores áreas de aplicación debido a que los dispositivos no necesitan tener el destino dentro de su rango de cobertura ya que la información se puede encaminar a través de un enlace de varios saltos hasta el destino. Este tipo de topologías resultan de gran interés en diferentes aplicaciones como el control y la monitorización para entornos industriales, la implementación de redes de sensores inalámbricas, redes destinadas a la agricultura o monitoreo estructural para edificios y puentes.

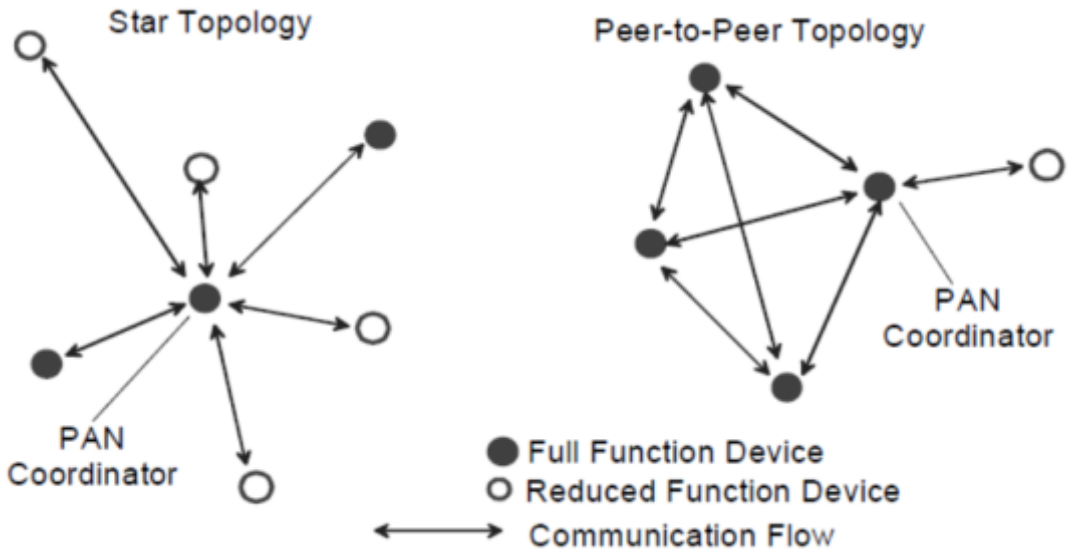


Figura 1. Ejemplos de topología en estrella y peer-to-peer

En el caso de la topología en estrella únicamente puede haber un coordinador que se encargará de gestionar la red PAN con un identificador único, y que trabajará de manera independiente a otras redes en estrella. En el caso de la topología peer-to-peer puede haber más de un dispositivo coordinador que proporcione una sincronización a otros dispositivos, pero solo uno de ellos será el coordinador de la red PAN. Este tipo de topologías tiene como ventaja que permite cubrir mayor área de cobertura, pero tiene el inconveniente de que se incrementa la latencia de los mensajes ya que en ocasiones tiene que ser reenviado.

2.1.1.1 Capa PHY

Las características de la capa física tienen que ver con la activación y desactivación del transceptor radio, la detección de energía, los indicadores de calidad del enlace (LQI), la selección de canal, la verificación de que el canal esté desocupado mediante el CCA (clear channel assessment) que se utilizará en el algoritmo CSMA-CA, así como la transmisión y la recepción de paquetes a través del medio físico.

La capa física del estándar IEEE 802.15.4 soporta diferentes bandas de frecuencias, así como diferentes tipos de modulaciones, siendo las más utilizadas las que se muestran a continuación:

- Banda de **2450 MHz** repartida en 16 canales de 5 MHz, utilizando una modulación del tipo O-QPSK.
- Banda de **915 MHz** repartida en 10 canales de 2 MHz, utilizando una modulación de tipo BPSK.
- Banda de **868 MHz** con un único canal de 2 MHz, utilizando una modulación de tipo BPSK.

Todas estas frecuencias utilizan una técnica de espectro ensanchado denominada DSSS (Direct Sequence Spread Spectrum) que en términos generales se utiliza para modular digitalmente una portadora, de tal forma que se aumente el ancho de banda de la transmisión y se reduzca la densidad de potencia espectral. De esta forma el espectro adquiere una forma parecida a la del ruido, permitiendo que solo pueda ser decodificado por el receptor que comparta la secuencia.

Entre las bandas de frecuencia mencionadas, existen un total de veintisiete canales, numerados desde k=0 hasta k=26. Se distribuyen de tal forma que la banda de 868 MHz tenga un solo canal, diez canales para la banda de 915 MHz y los dieciséis para la banda más alta de 2.4 GHz. Sus frecuencias centrales se obtienen mediante la Ecuación 2.1:

$$F_C = \begin{cases} 868.3\ en\ MHz,\ \ para\ k = 0 \\ 906 + 2(k-1)\ en\ MHz,\ \ para\ k = 1, 2, \dots, 10 \\ 2405 + 5(k-11)\ en\ MHz, para\ k = 11, 12, \dots, 26 \end{cases} \quad \textbf{(2.1)}$$

En la Figura 2 se pueden observar una representación de los veintisiete canales mencionados.

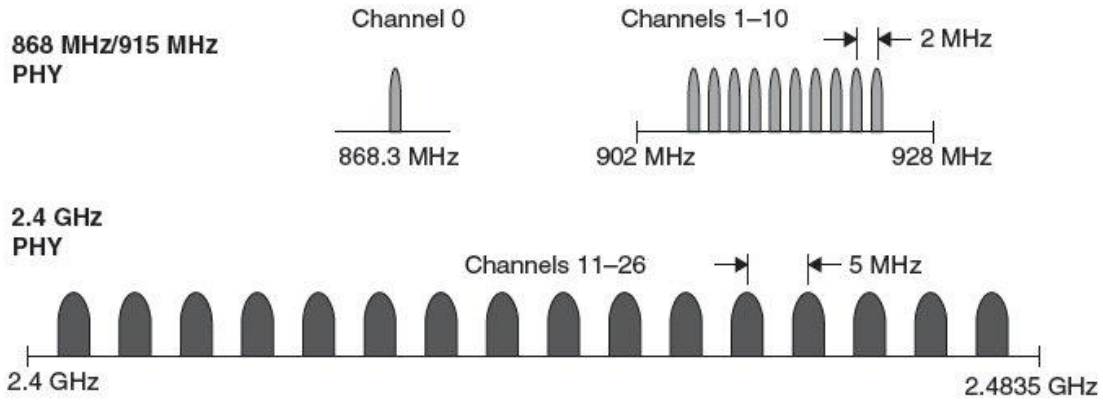


Figura 2. Bandas de frecuencias más utilizadas en IEEE 802.15.4

Aunque estos son los canales más utilizados, el estándar contempla otras bandas de frecuencias pensadas para ser utilizadas conforme a las normativas de algunos países asiáticos o simplemente para otro tipo de aplicaciones. Algunas de estas bandas son las de 433 MHz, 780 MHz, 2380 MHz o UWB (Ultra Wide Band) que pueden ser utilizadas junto con otros tipos de modulaciones y técnicas de espectro ensanchado para diferentes tipos de aplicaciones.

2.1.1.2 Capa MAC

Esta subcapa es la que permite controlar el acceso al canal radio mediante diferentes tipos de mecanismos, además de proporcionar dos servicios: el servicio de datos MAC y la administración de servicios MAC que interconecta con la entidad de administración de capa MAC (MLME – MAC sublayer management entity). El servicio de datos MAC es el que permite la transmisión y recepción de MPDUs (MAC protocol data units) a través del servicio de datos de la capa física.

Entre las características que permite esta capa se encuentran la administración de las beacons, el acceso al canal, administración de slots temporales garantizados (GTS), validación de tramas, retransmisión de mensajes de verificación, asociación y disociación. Además, se proporcionan las herramientas necesarias para implementar mecanismos de seguridad a este nivel.

En la versión del estándar anterior a la enmienda de 2012 estaban definidos dos métodos diferentes de acceso al canal, estos modos estaban caracterizados por si utilizaban o no las beacons a la hora de estructurar la supertrama, de ahí sus nombres beacon enabled (BE) y non-beacon enabled (NBE). En el modo beacon enabled la supertrama está delimitada por beacons en sus extremos, este formato estará definido por el coordinador de la red, que será el encargado de enviar esta estructura para que el resto de dispositivos se sincronicen. Opcionalmente, la supertrama puede estar dividida en una parte activa y otra inactiva, tal y como muestra la Figura 3. Esta porción de inactividad permite implementar un mecanismo de administración de potencia basado en el ciclo de trabajo de los dispositivos, ya que durante este periodo de inactividad los dispositivos pueden entrar en un modo de baja potencia para ahorrar energía. La transmisión de la beacon comienza al principio del primer slot de cada supertrama y se utiliza para la sincronización de los dispositivos enlazados, identificar la red mediante el PAN-ID y de describir la estructura de las supertramas.

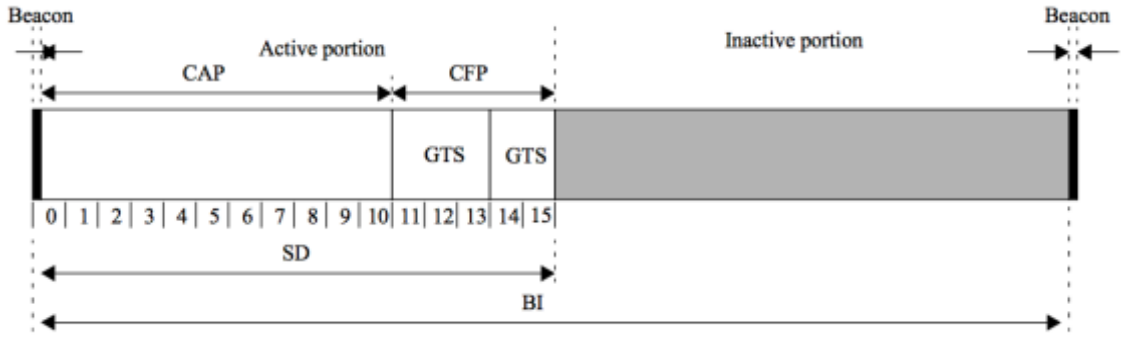


Figura 3. Ejemplo de supertrama

Dentro de la zona activa de la supertrama los dispositivos pueden establecer una comunicación entre ellos durante el periodo de acceso mediante contienda, denominado en inglés como Contention Access Period (CAP). Durante este periodo los dispositivos compiten entre ellos utilizando el mecanismo CSMA-CA ya comentado, aunque también está contemplado utilizar otras opciones como ALOHA. Para aquellas aplicaciones que requieran una baja latencia o requisitos de ancho de banda específicos, el coordinador de la red dedica una porción de este periodo activo para disponer de unos slots que formen un periodo libre de contienda, denominado en inglés como Contention-Free Period (CFP). Este periodo se divide en slots temporales garantizados (GTSs) que aparecen siempre al final de la zona activa. Las longitudes

de estos periodos CAP y CFP se puede ajustar dependiendo de los requisitos que tenga la aplicación en cuestión.

Para el modo NBE en el que no se utilizan las beacons para llevar una sincronización entre los dispositivos, el método de acceso al medio se realiza de manera aleatoria mediante el algoritmo CSMA-CA, al igual que en el periodo CAP de la supertrama del modo BE. Este mecanismo de acceso aleatorio comienza por comprobar si el canal está ocupado utilizando el método CCA de la capa física, y en el caso de que el canal esté ocupado esperará un tiempo aleatorio hasta volver a intentar la transmisión.

El número máximo de retransmisiones es 3. Esto significa que, para paquetes que requieren de ACK de comprobación, si no se recibe ninguno después de un total de 4 intentos, la transmisión se considera fallida y se debe notificar a los niveles superiores. Los paquetes que no requieran ACK de comprobación, entre los que se incluyen las beacons, no se retransmitirán.

2.1.1.3 Limitaciones de 802.15.4

El funcionamiento del protocolo de capa MAC 802.15.4 ha sido ampliamente investigado, tanto en su modo BE como el NBE, en una gran variedad de artículos, algunos de los cuales vienen recogidos en el survey [9]. De estos estudios se han podido derivar algunas de las limitaciones que el estándar presentaba antes de su actualización en 2012 y han servido para optimizar algunos de estos modos de funcionamiento. Algunas de estas limitaciones son las siguientes:

- Delay no acotado. En los diferentes métodos de acceso al canal de la capa MAC, al utilizar el algoritmo CSMA-CA, provocaba que fuese difícil estimar un máximo valor del delay hasta que la información alcanzase su destino.
- Fiabilidad limitada en la comunicación. La capa MAC de 802.15.4 utilizando el modo BE proporcionaba un valor muy bajo de tasa de entrega, incluso cuando el número de nodos no era muy alto. Esto se debe principalmente a la ineficiencia del algoritmo CSMA-CA ranurado utilizado para el acceso al canal. Lo mismo ocurre en el caso del modo NBE, en el que un gran número de nodos puede empezar a transmitir simultáneamente.
- No existe protección frente a interferencias/desvanecimientos. Las interferencias y los desvanecimientos multi camino son fenómenos muy comunes en las redes inalámbricas. El estándar de 802.15.4, al utilizar un único canal y no implementar mecanismos de salto en frecuencia, no permite mitigar dichos efectos negativos. Por esto, la red es propensa a la inestabilidad e incluso a perder la conexión de sus nodos.
- Consumo de nodos retransmisores. El estándar soporta tanto topologías en estrella como topologías multi salto (peer-to-peer). De esta forma es posible crear topologías en la que los nodos intermedios no necesiten permanecer todo el tiempo activo, sin embargo, en estos métodos de acceso al medio se requieren complejos mecanismos de sincronización y planificación de beacons que no vienen especificados por el estándar. Para superar estas limitaciones, muchas de las aplicaciones mantienen los dispositivos retransmisores activos todo el tiempo, causando así un gran consumo de energía.

Por estas razones, las primeras versiones del estándar IEEE 802.15.4 no resultaba muy útil en escenarios donde la fiabilidad y la robustez sean requisitos estrictos, como puede ser el caso de las aplicaciones en escenarios industriales.

2.1.1.4 Protocolos propietarios basados en IEEE 802.15.4

Debido a estas limitaciones que presentaba el estándar, diferentes organizaciones industriales como HART Communication Foundation, ISA, WINA o ZigBee han estado trabajando activamente para mejorar las aplicaciones y tecnologías inalámbricas en el sector de la automatización industrial. Estos grupos notaron que el estándar de 802.15.4 tenía ciertas carencias para satisfacer los requisitos fundamentales en aplicaciones de ámbito industrial, ya que con los métodos de acceso al medio que se contemplaban en ese momento no se podían garantizar dichos requisitos de fiabilidad y robustez de las comunicaciones. A continuación, se presentan los cuatro estándares para WSN industriales más populares realizando una breve comparación entre sus características principales [10]. Todos ellos se basan en el estándar de IEEE 802.15.4, realizando algunas modificaciones a nivel de MAC y completando la pila de protocolos para obtener una plataforma completa.

WirelessHART

Figura 4. WirelessHART logo.

WirelessHART [11] es una extensión del protocolo HART (Highway Addressable Remote Transducer) capaz de soportar comunicaciones inalámbricas. El estándar WirelessHART ha sido la primera tecnología de WSN para el entorno industrial que alcanza el nivel de reconocimiento internacional. WirelessHART se encarga de construir una red mallada inalámbrica y está diseñado en base a una serie de requisitos fundamentales: debe ser simple, auto-organizable, auto-reparable y flexible, escalable, fiable, segura y capaz de soportar la tecnología HART existente. Fue aprobado como estándar público por la International Electrotechnical Comission en 2009, actualizada en 2010 y su última actualización ha sido en abril de 2016.

La pila de protocolos de WirelessHART comprende cinco capas: la capa PHY que se basa en el estándar IEEE 802.15.4, la capa de Enlace (que incluye las funciones de MAC), la capa de Red, la capa de Transporte y la capa de Aplicación.

ISA100.11a

Figura 5. ISA 100 logo.

ISA 100.11a [12] fue desarrollada por la International Society of Automation (ISA) y posteriormente fue estandarizado por la IEC (IEC 62734), cuya última actualización fue en 2014. ISA100.11a define la pila de protocolos, la administración de sistema y funciones de seguridad para ser utilizadas en redes inalámbricas de baja-potencia y baja-tasa. Las capas de red y de transporte están basadas en los estándares 6LoWPAN, IPv6 y UDP. La capa de enlace de datos es una de ISA100.11a ya que usa algunas modificaciones de la capa MAC de IEEE802.15.4 para añadir características de graph routing, frequency hopping y TDMA de las que carecía la primera versión de 802.15.4.

ZigBee Pro

El protocolo ZigBee [13] es otra de las especificaciones basadas en el estándar IEEE 802.15.4 para WSN, destinada a la radiodifusión digital de bajo consumo. Este estándar construye los niveles superiores de la pila de protocolos a partir del nivel MAC, que hereda de 802.15.4 junto con la capa física. Su propósito es ofrecer una solución completa para poder implementar una WSN hasta el nivel de aplicación.

En su origen, el protocolo ZigBee era un sistema ideal para ser utilizado en redes domóticas y fue creado por la ZigBee Alliance para cubrir la demanda del mercado de un sistema de bajo coste, un estándar para redes inalámbricas de pequeños paquetes de información, bajo consumo, seguro y fiable.

Posteriormente, lanzaron la versión de ZigBee PRO que mejoraba su rendimiento en entornos más críticos, ya que estaba más enfocado a dar un servicio en aplicaciones industriales. La primera de las versiones utilizaba una frecuencia única para establecer las comunicaciones de toda la red, siendo esto susceptible a interferencias y desvanecimientos. Con la nueva versión PRO se implementaron algunas características que permitían tener una flexibilidad a la hora de seleccionar la frecuencia de comunicación, permitiendo cambiar si el canal no tenía la suficiente calidad. En los últimos años se han añadido además nuevas funcionalidades que permiten implementar protocolos como IPv6 o mecanismos de encaminamiento como RPL y combinarlas con las especificaciones de ZigBee PRO, ya que estos protocolos están siendo adoptados como pilares fundamentales en el desarrollo de WSNs y el Internet de las Cosas.

WIA-PA

WIA-PA (Wireless network for Industrial Automation – Process Automation) [14] ha sido desarrollado por la Chinese Industrial Wireless Alliance, posteriormente fue aprobado por la IEC como estándar internacional en el año 2010. Actualmente, solo es posible adquirir productos certificados para WIA-PA en China por lo que resulta complicado acceder a la información relevante sobre este estándar. Permite construir una topología que combina arquitectura en estrella y en malla de forma que el primer nivel se distribuyan los nodos en una topología mallada y en el segundo nivel en estrella. Estas estructuras se denominan cluster y se utilizan para crear un sistema de intercambio de información entre dispositivos dentro del mismo cluster y entre distintos cluster.

Arquitectura de red

En cuando a la arquitectura de las redes formadas por los distintos protocolos industriales, podemos decir que en esencia siguen una misma línea de red jerarquizada y centralizada, controlada por un sistema de control externo a la red de sensores inalámbrica. El que más se aleja de esta estructura es el protocolo ZigBee PRO, que hereda su arquitectura del ZigBee tradicional, un sistema controlado por un nodo coordinador que forma parte de la red de sensores y que mantiene el estado de los enlaces que forman toda la red. Por otro lado, el resto de protocolos opta por dirigir toda la información que circula por la red hacia un sistema ajeno a la red inalámbrica, que puede estar conectado de manera cableada al Gateway, en el que desembocan los datos de la red. Este sistema, denominado Network/System Manager por los protocolos, puede estar dividido además en diferentes roles, encargados por un lado de la configuración de la red, el control de la topología, el estado de los enlaces o la gestión de recursos radio y por otro lado de la gestión de la seguridad o accesibilidad de los nodos. Toda esta información fluye hasta estas estaciones centrales, controlando la salud de los enlaces y la formación de toda la red.

En cuanto a los tipos de dispositivos que forman la red inalámbrica, WirelessHART contempla únicamente unos dispositivos de campo desplegados en planta que tienen implementadas una funcionalidad completa, con capacidades de enrutamiento en todos ellos. Además, conectados al Gateway coloca uno o varios puntos de acceso que se encarguen de retransmitir los datos de la red hacia el Gateway.

El resto de protocolos sí que permite configurar los nodos de tal forma que puedan servir como nodos intermedios capaces de encaminar la información o simplemente como un dispositivo de campo que sirva de nodo sensor/actuador. Además, hay un tipo de nodos denominados Handheld Device que sirven a los operarios de planta para ir desplazándose y poder ir configurando diferentes aspectos de la red de sensores. Estos nodos actúan como un dispositivo más de la red, pero sus capacidades de movilidad y de administración de la red lo hacen muy útil como dispositivo de configuración.

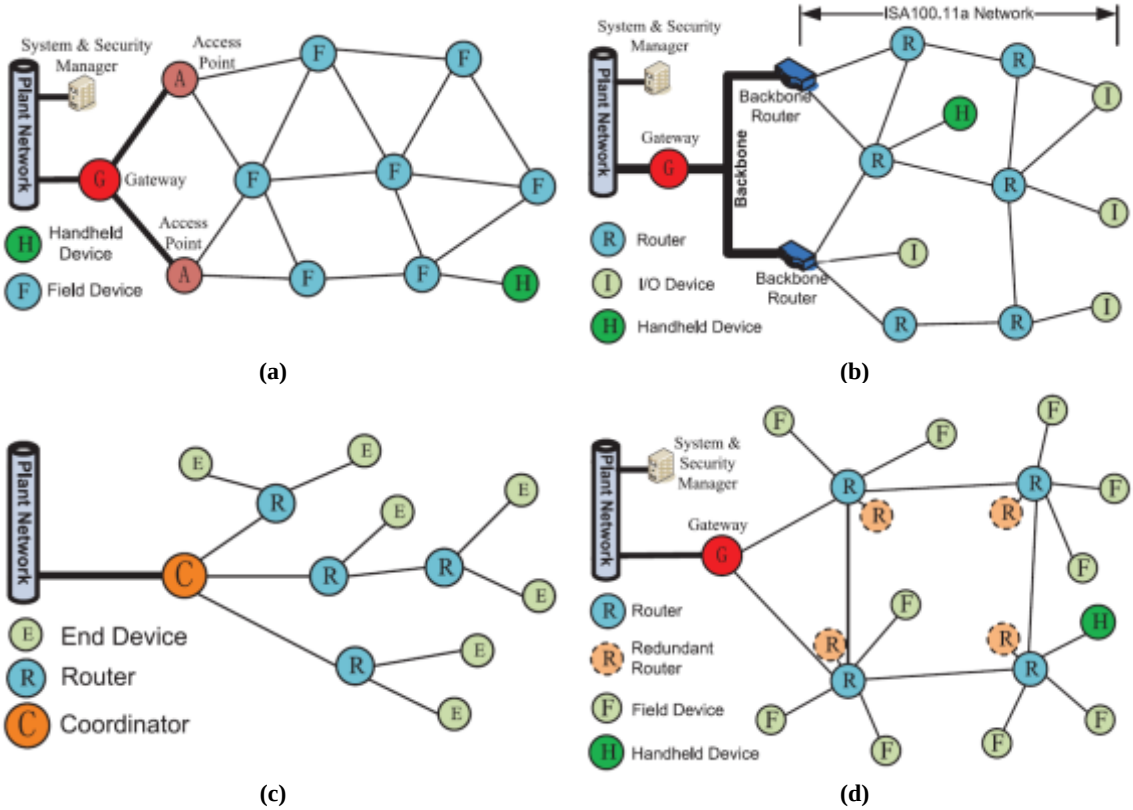


Figura 6. Arquitectura de red de (a) WirelessHART, (b) ISA 100.11a, (c) ZigBee y (d) WIA-PA [10].

Arquitectura de protocolos

En cuanto a la pila de protocolos que implementa cada uno de estos estándares, podemos ver en la Figura 7 como se distribuyen los diferentes niveles sobre la definición del estándar IEEE 802.15.4.

El protocolo ZigBee adopta la capa MAC completa de 802.15.4 sin realizar ninguna modificación, implementando la funcionalidad de poder seleccionar el canal de transmisión en el que estará funcionando toda la red, pero sin permitir que los nodos vayan cambiando este canal de manera flexible.

Por otro lado, WirelessHART, ISA100.11a y WIA-PA coinciden en que la capa MAC de 802.15.4 no es lo suficientemente fiable para implementarlo en aplicaciones industriales por lo que deciden añadir sus propias características para solucionarlo. Por un lado, WIA-PA opta por renovar toda la estructura, pero dejando intactas las capas definidas por 802.15.4, por lo que añade características como Frequency Hopping, Packet Aggregation and Disaggregation y Time Synchronization en una capa adicional de datos de enlace (DLL), aunque estas características son propias de la capa MAC.

WirelessHART e ISA100.11a prefieren eliminar algunas de las características de la capa MAC y definir sus propias funciones de Channel Hopping, Slot Timing Communications y Time Synchronized TDMA/CSMA.

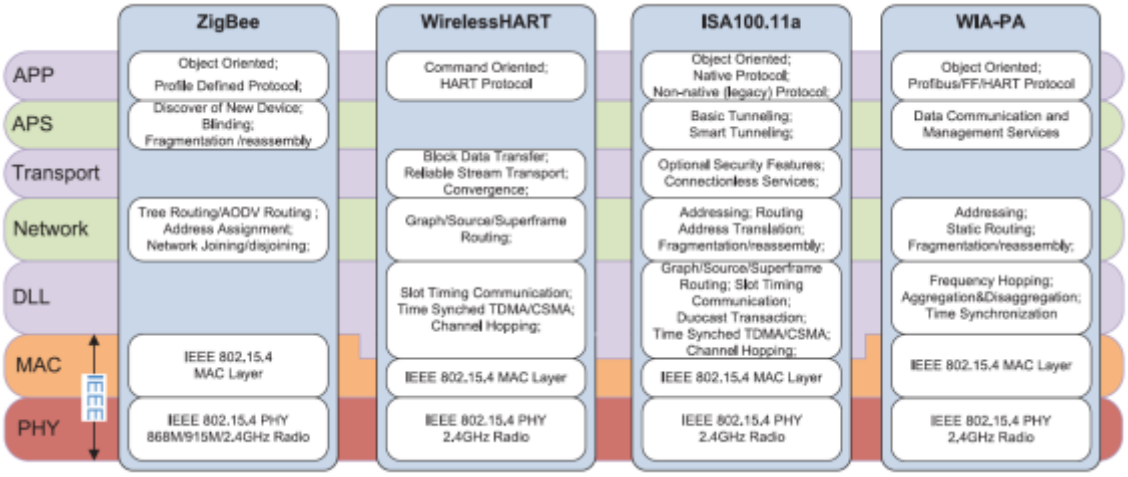
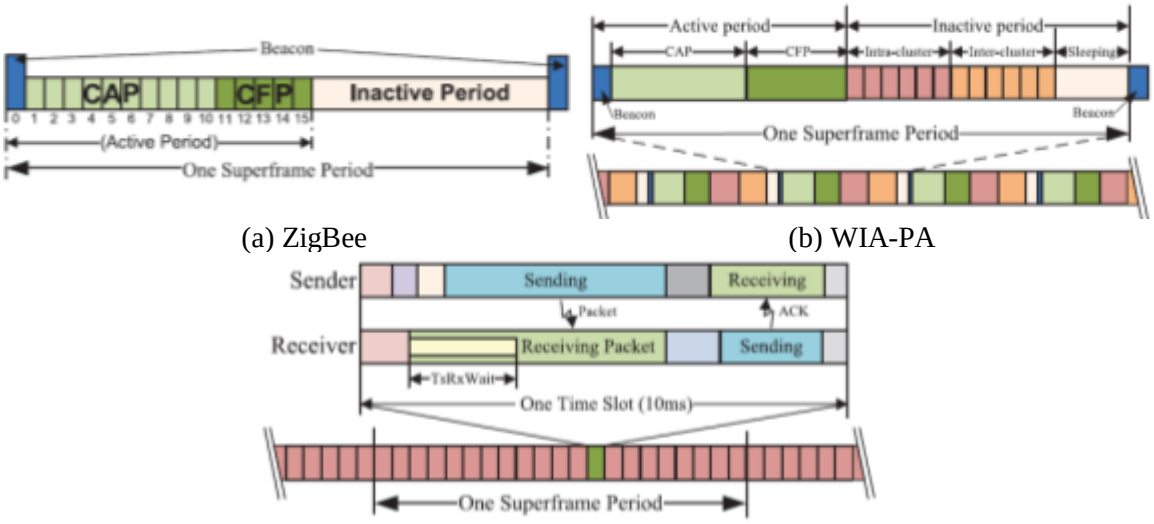


Figura 7. Pila de protocolos de estándares industriales [10]

En cuanto al uso de tramas, la Figura 8 recoge las diferentes aproximaciones que realiza cada uno de los protocolos industriales. ZigBee utiliza la estructura limitada por dos Beacons definida por el estándar IEEE 802.15.4 y denominada Supertrama, que se divide en diferentes periodos para acceder a los recursos de manera dedicada o bien entrando en contienda con otros dispositivos para adquirir el recurso compartido.

WIA-PA modifica esta definición de Supertrama, utilizando el periodo de CAP para la unión de nuevos dispositivos, la administración de grupos de nodos que denomina cluster o para los reintentos, mientras que el CFP lo utiliza para establecer las comunicaciones entre los dispositivos y el nodo coordinador. Además, reduce el periodo de inactividad para incluir unos periodos que utiliza para establecer comunicaciones intra-cluster e inter-cluster.

Por último, WirelessHART e ISA100.11a sustituyen la Supertrama que define el estándar IEEE 802.15.4 por un conjunto de slots temporales que se repiten en el tiempo, denominado slotframe. Estos slots tendrán una duración fija de 10 ms en el caso de WirelessHART, tiempo suficiente para que un nodo transmita un paquete y pueda recibir el ACK de comprobación, aunque en ISA100.11a esta duración también es configurable dando mayor flexibilidad al protocolo.



(a) ZigBee (b) WIA-PA

(c) WirelessHART e ISA100.11a

Figura 8. Estructura de Supertrama y Slotframe de protocolos industriales.

Las características que implementan estos protocolos para poder realizar saltos en frecuencia, consiguen mitigar las interferencias y los desvanecimientos debidos al efecto multi-camino ya que las comunicaciones van modificando su canal de operación. Esta característica resulta muy interesante y es por esto que el estándar IEEE 802.15.4 recoja las diferentes propuestas de estos protocolos para crear un documento de enmienda que permita implementar estas características de manera nativa en 802.15.4. Por ello se describirá su funcionamiento más adelante.

2.1.1.5 Enmienda IEEE 802.15.4e

La última actualización del estándar IEEE 802.15.4 es la enmienda IEEE 802.15.4e que se creó con el objetivo de rediseñar la capa MAC del estándar para solventar algunas de sus limitaciones. El objetivo era definir un protocolo a nivel de MAC que permitiera crear redes multi-salto de baja potencia que fuese capaz de cumplir con las necesidades emergentes de las aplicaciones industriales. El resultado final fue el documento *IEEE 802.15.4e MAC Enhancement Standard* aprobado en el año 2012, que recogía muchas ideas de los estándares industriales ya existentes, WirelessHART o ISA 100.11a [3], como el acceso a los slots temporales, celdas compartidas y dedicadas, establecimiento de comunicaciones multi-canal y la implementación de patrones de salto frecuencial.

La modificación de IEEE 802.15.4e introduce algunas mejoras en el funcionamiento general del estándar, que podemos resumir en los siguientes puntos.

- **Mecanismo Low Energy (LE).** Este método está pensado para aquellas aplicaciones en las que se priorice la eficiencia energética a costa de empeorar la latencia. De esta forma se permite que los nodos funcionen con un ciclo de trabajo muy bajo, alrededor del 1%, mientras que para los niveles superiores el nodo permanece como que siempre está activo.
- **Information Elements (IE).** Es un mecanismo añadido para el intercambio de información mediante la capa MAC. Estos Information Elements no son más que un conjunto de cabeceras que permiten encapsular distintos tipos de información para transmitirla en los paquetes a nivel de MAC.
- **Enhanced Beacons (EB).** Son una extensión de las beacons tradicionales del estándar 802.15.4 que proporcionan mayor flexibilidad manteniendo la compatibilidad con el resto. Esto permite crear tramas para aplicaciones específicas en las que las beacons transporten información contenida en los Ies.

En cuanto a los modos de acceso al medio, se incluyen los siguientes cinco métodos que mejoran el acceso al medio.

- El modo **DSME** (*Deterministic & Synchronous Multi-channel Extension*): se ha diseñado para aplicaciones cuyos requisitos sean alta disponibilidad, eficiencia, escalabilidad y robustez soportando tanto aplicaciones industriales como comerciales. Para ello combina el acceso mediante contienda y la división temporal del medio de acceso, ofreciendo además dos modos diferentes de diversidad de canales.
- El modo **LLDN** (*Low Latency Deterministic Network*): desarrollado para aplicaciones que requieren muy poca latencia, como robots, grúas, etc. Está pensado para redes de un único salto y de un único canal.
- El modo **TSCH** (*Time Slotted Channel Hopping*): enfocado principalmente para entornos industriales donde el consumo de energía tiene que ser reducido y la diversidad y robustez frente a interferencias tiene que ser alta.
- El modo **RFID Blink** (*Radio Frequency Identification Blink*): diseñado para la comunicación con dispositivos aparte de identificadores que estos incorporan. Se usa para aplicaciones de identificación de objetos o personas, localización o seguimiento de los mismos.

- El modo **AMCA** (*Asynchronous Multi-Channel Adaptation*): es usado en redes de gran tamaño y dispersión geográfica tales como las redes para Smart Utility, redes de monitorización de infraestructuras y redes de control de proceso.

Los últimos dos modos se describen brevemente por el estándar y su presencia en artículos de investigación es bastante reducida, dejando los modos TSCH, DSME y LLDN como los más interesantes y característicos de este estándar. En la siguiente tabla se recogen las principales características de estos tres modos.

	TSCH	DSME	LLDN
Beacons	Si (Enhanced Beacons)	Si (Enhanced Beacons)	Si
Organización temporal	Slotframe periódico: - Número arbitrario de slots temporales. - Los slots pueden ser dedicados o compartidos.	Multisuperframe periódico: - Estructura bien definida. - Utiliza los periodos de CAPs y CFPs de la trama.	Superframe periódico: - Se definen 3 estados de transmisión. - Administración, uplink, slots bidireccionales. - Está pensado para slots temporales cortos (< 1 ms)
Acceso al canal	- División temporal en slots (para los slots dedicados) - TSCH CSMA-CA (para los slots compartidos)	- Acceso basado en contienda durante el periodo de CAPs. - División temporal en slots durante el periodo de CFPs.	- División temporal en slots (para los slots dedicados) - LLDN CSMA-CA (para los slots compartidos)
Topologías	Estrella, árbol, malla	Estrella, árbol, malla	Estrella
Mecanismo multi-canal	Channel hopping	- Channel hopping - Channel adaptation	No tiene mecanismos
Mecanismo de planificación de slots temporales	No se especifica	Asignación de slots garantizados (GTS) de manera distribuida.	Centralizada.
Sincronización de la red	Basada en la transmisión/recepción de parejas trama/ACK	Recepción de las Enhanced Beacons	Recepción de las beacons

Tabla 1. Principales características de TSCH, DSME y LLDN

2.1.2 Time Synchronize Channel Hopping (TSCH)

El método de acceso al medio mediante TSCH es un modo principalmente pensado para soportar aplicaciones de automatización de procesos, con un enfoque en particular en la monitorización de equipamiento y procesos, lo que lo hace un método idóneo para aplicaciones en escenarios industriales. Algunos de los dominios de aplicación de este modo de operación son la industria de gas y petróleo, productos alimenticios y bebidas, productos químicos y farmacéuticos, tratamiento de aguas residuales, producción de energía limpia o sistemas de control climático [8].

En este modo de operación se sustituye el concepto de supertrama, como la que se ha descrito hasta ahora en el estándar, por una trama dividida en slots temporales denominada slotframe. Los slots por los que está formado el slotframe pueden ser, periodos de comunicación entre pares de dispositivos en el que se garantice el recurso o es posible acceder a ellos de manera aleatoria mediante CSMA-CA. Los slots estarán repetidos en el tiempo de manera periódica y compartirán una noción de tiempo entre los dispositivos participantes. Ya que todos los dispositivos comparten un origen de tiempos común y una información sobre el grupo de canales utilizados, la comunicación en varios saltos se realiza utilizando el espacio completo de canales para así minimizar los efectos negativos de las interferencias y los desvanecimientos debidos al efecto multi-camino. Además, la utilización de una trama de slots temporales puede permitir que se eviten las colisiones organizando las transmisiones para que se produzcan en instantes de tiempo diferentes, minimizando la necesidad de retransmitir los datos. Es por estas características que su modo de operación resulta tan interesante en los escenarios industriales.

Por tanto, el modo TSCH proporciona un incremento de la capacidad de la red, una alta fiabilidad y una latencia predecible mientras que mantiene un ciclo de trabajo muy bajo gracias al método de acceso por slots temporales. Además, funciona independientemente del tipo de topología que se utilice, por lo que se puede utilizar en redes configuradas en estrella, árbol o cualquier tipo de variación de redes malladas. Es particularmente útil en redes multi-salto ya que su característica de salto en frecuencia permite realizar un uso eficiente de los recursos disponibles en la red.

2.1.2.1 Estructura de los slotframes y sincronización

Como ya se ha comentado, un slotframe es un conjunto de slots temporales repetidos en el tiempo. Cada uno de estos slots dura el tiempo suficiente para que un par de dispositivos puedan intercambiar una trama junto con su paquete ACK de reconocimiento. Este tamaño de slot es configurable, lo que permite adaptarlo para aplicaciones en los que los mensajes sean más largos o que requieran de otra temporización. El número de slots que se incluyen en un slotframe (tamaño del slotframe) determina la cantidad de veces que un slot se repite, de manera que los nodos pueden planificar que slots van a utilizar. Cuando se crea un slotframe, se le asocia un valor denominado macSlotframeHandle que permite identificar el slotframe, y cada uno de ellos se podrá repetir con un ciclo diferente dependiendo de su longitud. La configuración de los slotframe y de los slots temporales se realiza por los niveles superiores.

Para poder identificar los slots que han transcurrido desde el inicio, se define un valor llamado Absolute Slot Number (ASN) que indica el número de slots que han transcurrido desde el inicio de la red o de un tiempo de inicio arbitrario determinado por el coordinador de la red PAN. Este valor se incrementa globalmente en la red cada vez que transcurre un slot temporal, tiempo que viene definido por el parámetro macTsTimeslotLength. Este valor de ASN deberá de transmitirse en las beacon de señalización por los dispositivos que ya pertenezcan a la red TSCH para que nuevos dispositivos puedan unirse y sincronizarse. Además, este valor se utiliza para calcular el canal para cualquier comunicación entre dos pares de dispositivos.

En la Figura 9 se muestra un ejemplo de comunicación entre nodos con un slotframe compuesto por tres slots temporales. En cada uno de estos slots estará configurado un enlace en el que se establecerá la comunicación entre un par de dispositivos. Por ejemplo, en el slot TS0 el nodo A se comunica con el nodo B y en el TS1 se comunica el B con el C. Cada vez que transcurren 3

slots temporales, la planificación se repite y es importante resaltar que el valor del ASN se incrementa continuamente independientemente de las repeticiones de los enlaces planificados. La asignación de una comunicación entre dos dispositivos vendrá dada por el slot temporal que estén planificados y el channel offset del enlace.

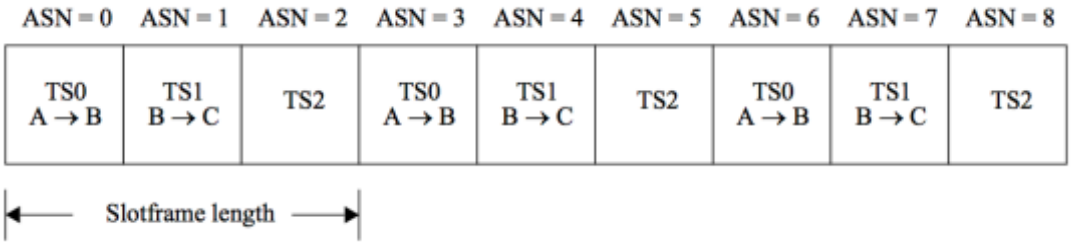


Figura 9. Ejemplo de un slotframe con tres slots temporales.

El canal físico de transmisión (CH) en un enlace se calcula a partir del valor de ASN y del channel offset en el que esté planificado el enlace, y dependerá también del número total de canales que haya en la secuencia de salto. El canal físico vendrá dado por la Ecuación

$$CH = HoppingSeqList[(ASN + ChOffset)\ \%\ HoppingSeqLength] \quad \textbf{(2.2)}$$

donde a%b indica a módulo de b.

Utilizar un channel offset permite que diferentes canales, de la lista de patrones de salto de canales, puedan ser utilizados para un valor de ASN dado. Hay tantos channel offset como canales, lo que resulta en un único canal para una combinación de ASN y secuencia de salto. Siguiendo el ejemplo de la Figura 9, podemos suponer que todo el slotframe está planificado en un channel offset igual a cero. Suponemos que tenemos también una secuencia de salto de cuatro canales como la siguiente:

$$HoppingSeqList = \{15, 25, 26, 20\}$$

Si utilizamos la Ecuación 2.2 podremos comprobar que el slot TS0 utilizará el canal 15 para transmitir en el ASN = 0, pero en el ASN = 3 en el que vuelve a repetirse el slot TS0 se utilizará el canal 20. Esto se debe a una de las características de la longitud del slotframe, que deberá ser de un número de slots primos de longitud para que así, un mismo slot pueda transmitir por todos los canales de la secuencia de salto.

El estándar contempla la posibilidad de poder utilizar varios slotframes de manera simultánea, pudiendo tener estos diferentes tamaños. Esto puede servir para definir diferentes planificaciones de comunicación para grupos de diferentes nodos o para que la red funcione con diferentes ciclos de trabajo dependiendo de si los dispositivos utilizan un slotframe u otro. Un mismo dispositivo puede participar en uno o más slotframe de manera simultánea, lo que permitiría establecer diferentes planificaciones y matrices de conectividad que funcionen al mismo tiempo.

Estos slotframes pueden añadirse, eliminarse o ser modificados mientras que la red está en tiempo de ejecución, aunque todos ellos estarán alineados por unos límites que vendrán determinados por el coordinador de la red PAN, ya que los dispositivos deben de seguir sincronizados. De esta forma, los slots temporales de diferentes slotframes estarán siempre alineados, incluso cuando no tengan el mismo tamaño. Esta característica está ilustrada en la Figura 10. En el caso de que un dispositivo tenga enlaces planificados en varios slotframes y que coincidan en un mismo instante de tiempo, las transmisiones tienen precedencia frente a las recepciones y los slotframes que estén identificados por un valor macSlotframeHandle menor, tendrán preferencia frente a los que tengan un valor mayor. Esta característica de utilizar diferentes slotframes es explotada por aplicaciones como Orchestra para construir las planificaciones en las redes TSCH.

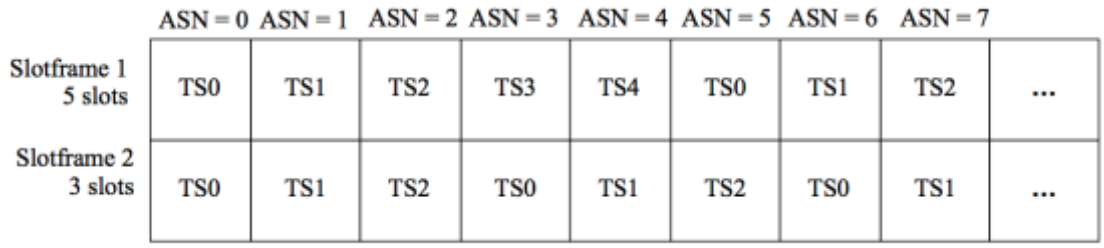


Figura 10. Múltiples slotframes en una red TSCH.

2.1.2.2 Algoritmo TSCH CSMA-CA

En aquellos slots que permitan establecer enlaces compartidos, los dispositivos podrán acceder a ellos de forma simultánea, por lo que si se produce una colisión podrá acabar en una transmisión fallida. Para reducir la posibilidad de que se vuelva a producir otra colisión, el método de acceso TSCH utiliza el algoritmo CSMA-CA para las retransmisiones.

Cuando un dispositivo tiene un paquete disponible para enviar a un destino, el emisor espera la llegada del primer enlace en el que esté planificada la comunicación entre dicho emisor y receptor, para así poder transmitir los datos que tiene preparados. Si la transmisión se realiza a través de un enlace compartido y resulta que la comunicación no se finaliza con éxito (ya sea porque no ha llegado el mensaje o no se recibe el ACK de verificación), es muy probable que se haya producido una colisión. Cuando esto sucede el nodo ejecuta el algoritmo CSMA-CA para evitar que se repita la colisión, para ello el nodo sigue los siguientes pasos:

1. Se generan un conjunto de variables, una es el número de retransmisiones que se ha realizado para el paquete que no se ha transmitido con éxito (NB = 0) y el otro es una variable denominado exponente de backoff (BE = macMinBE).
2. Se genera un número aleatorio comprendido entre $[0, 2^{BE} - 1]$.
3. La retransmisión del paquete fallido se retrasa un número de slots compartidos igual al número aleatorio generado cuyo destino sea el correspondiente, o bien hasta encontrar un enlace dedicado que coincida con la destinación.
4. Si la retransmisión se realiza en un enlace compartido y resulta exitosa, el exponente de backoff BE se reinicia de nuevo a macMinBE y el algoritmo finaliza. Por otro lado, si la transmisión vuelve a fallar, las variables de estado se actualizan de tal manera que NB = NB + 1 y BE = min(BE+1, macMaxBE). Por último, si el número de retransmisiones de un mismo paquete supera el valor máximo permitido (macMaxFrameRetries) la trama se descarta.

Si la retransmisión de la trama se realiza en un enlace dedicado y resulta exitoso, el valor de BE se reinicia, a menos que haya otras tramas destinadas al mismo receptor y que estén preparadas para transmitir. En este último caso el valor de BE permanece constante.

2.1.2.3 Formación de la red TSCH

Una red TSCH se forma cuando un dispositivo notifica la presencia de la red enviando tramas de Enhanced Beacon. Estas tramas utilizadas como balizas contienen los siguientes Information Elements:

- TSCH Synchronization IE: contiene información temporal para que los nuevos dispositivos sean capaces de sincronizarse con la red.
- Channel hopping IE: contiene información relativa al patrón de salto de canales que se utilizará en la red.
- TSCH Timeslot IE: contiene información que describe las características del slot temporal. En él se indica cuando se espera transmitir una trama y cuando enviar un ACK de verificación.

- TSCH Slotframe and Link IE: contiene información sobre el slotframe y el enlace inicial para que los nuevos dispositivos sepan cuando deben estar escuchando para recibir la información sobre la red.

Los dispositivos que pretenden unirse a la red empiezan a escanear de manera pasiva o activa la red, en busca de recibir algún Enchanced Beacon. Una vez el dispositivo ha recibido un Enhanced Beacon valido, este se lo notifica a los niveles superiores para después seleccionar la red TSCH basándose en el campo Join Metric incluido en el TSCH Synchronization IE. Los niveles superiores son los que se encargan de inicializar el slotframe y los enlaces que vienen incluidos en el Enhanced Beacon y de cambiar al dispositivo en el modo TSCH.

Una vez llegados a este punto el dispositivo ya está sincronizado con la red y puede intentar asignar recursos adicionales de comunicación (slotframes y enlaces) con los dispositivos que ya están unidos. Una vez está unido, el dispositivo puede recibir información de los niveles superiores de administración para que incluya sloframes y enlaces nuevos o para que los elimine.

2.1.3 Algoritmos de planificación

Como hemos visto, el modo TSCH como método de acceso al medio mejora en gran medida el rendimiento de IEEE 802.15.4 ya que sus características consiguen hacer frente a las interferencias y los desvanecimientos. Sin embargo, en el estándar no se indica la manera óptima de gestionar los recursos radio que pone a nuestra disposición el modo TSCH. Se definen todas las características que deben tener los slots, la manera de poder configurarlos, acceder a ellos de manera compartida o dedicada, patrones de salto en frecuencia y toda una serie de métodos que hacen que TSCH resulte tan interesante en el despliegue de redes inalámbricas robustas. En última estancia, la manera en la que se reparten estos slots y canales dependerá del número de nodos, la extensión de la red o del tipo de tráfico que circule por la red, por lo que el estándar no puede definir una manera única de planificar los slots. Es por ello que la comunidad científica ha centrado sus esfuerzos en investigar diferentes tipos de algoritmos y métodos de planificación en los últimos años.

Estos métodos de planificación se pueden dividir en diferentes categorías en función de que organismo es el encargado de decidir la planificación de toda la red. Los primeros que podemos comentar son los algoritmos de planificación centralizados, en el que un único nodo central se encarga de recoger la información necesaria de los nodos que forman la red y construir una planificación según estos datos. Este tipo de algoritmos tiene el inconveniente de que es necesario realizar un gran intercambio de información para que un único nodo coordinador recoja toda la información útil para construir la planificación y una vez construida volver a transmitir a los diferentes nodos para que adopten dicha planificación.

Por otro lado, tenemos los algoritmos de planificación distribuidos. En estos, los nodos intercambian información con sus vecinos más cercanos y construyen la planificación en función de este intercambio de datos. De esta forma se consigue reducir el tráfico de datos que circula por la red, pero a costa de perder la vista global de la red. Es posible conseguir planificaciones libres de colisiones con algoritmos distribuidos, pero serán menos optimizados que el que conseguiría un algoritmo centralizado. De esta forma se deberá llegar a una situación de compromiso en el que primen las características que más nos interese mantener.

Un tipo particular de algoritmos distribuido son los llamados mecanismos autónomos de planificación. Este tipo de métodos consiguen crear una planificación local en cada uno de los nodos sin necesidad de intercambiar más información que la que ya se intercambian por otros protocolos. En este caso los nodos serían capaces de construir una planificación una vez que se han unido a una red de manera autónoma utilizando la información de la que ya dispone. Este es el caso de Orchestra, del que hablaremos más adelante.

2.1.3.1 Minimal 6TiSCH

Con el objetivo de plantear una configuración sencilla de planificación de recursos en TSCH, el grupo IETF ha proporciona la siguiente especificación [16] que plantea una configuración mínima de operación para proporcionar conectividad IPv6 a través de una red IEEE 802.15.4 formada por enlaces TSCH. En una red 6TiSCH los nodos siguen una planificación definida para TSCH, mientras que el direccionamiento de IPv6 y la compresión se consiguen a través de la estructura implementada por 6LoWPAN.

Este modo de operación mínimo utiliza un único slotframe TSCH que estará formado por un número variable de timeslots en el que haya solo una de estas celdas planificadas. El tamaño del slotframe se podrá modificar para llegar a una situación de compromiso entre ancho de banda y energía consumida, quedando esto fuera del ámbito de la RFC. Cuanto mayor sea el tamaño del slotframe tendremos un mayor número de celdas inactivas, lo que se traduce en que el nodo pasará más tiempo en un estado de bajo consumo, mientras que el ancho de banda será más reducido ya que la única celda activa tendrá un periodo de repetición mayor.

En la Figura 11 se muestra un ejemplo de trama de una longitud de 101 timeslots y un número de channel offset igual a 16. Como vemos solo el primer slot de la primera trama y channel offset cero es el que está activo, resultando en un ciclo de trabajo por debajo del 0.99%.

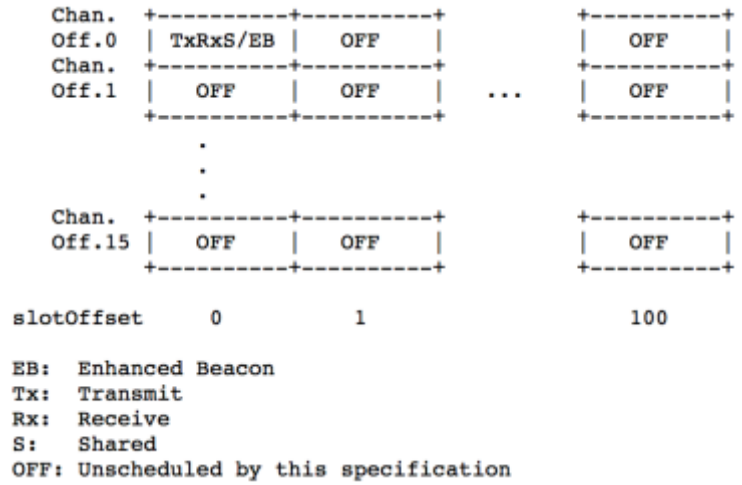


Figura 11. Ejemplo de Slotframe de longitud 101 Timeslots

Mediante esta configuración un nodo puede utilizar la celda planificada para transmitir/recibir cualquier tipo de trama, pudiendo utilizarlo también para transmitir las beacons de señalización. Además, esta planificación será común para todos los nodos, por lo que la celda estará compartida, necesitando por lo tanto resolver situaciones de contienda cuando varios nodos quieran acceder al recurso.

Esta configuración es la representación más simple de las posibles planificaciones TSCH, resultando insuficiente cuando el número de nodos que forma la red es bastante alto, ocasionando que el único recurso radio esté ocupado la mayor parte del tiempo y que algunos nodos no puedan acceder a él.

2.1.3.2 TASA

Traffic Aware Scheduling Algorithm (TASA) [17] [18] es un algoritmo centralizado, por lo que el coordinador de la PAN será el encargado de construir la planificación y enviársela al resto de nodos. Para esto necesita conocer cierta información sobre el estado de la red, en concreto:

- El grafo G (que representa la estructura en árbol).
- El grafo P (que representa la conectividad física entre los nodos).
- $\widetilde{q_i}$ con $i = 0$ (es el número de paquetes por slotframe generado por los otros nodos).

Utilizando estos datos, el algoritmo aplica dos conceptos conocidos en teoría de grafos como:

- **Matching** o emparejamiento: un matching en un grafo es un conjunto de aristas disjuntas, en este caso enlaces sin nodos en común.
- **Vertex coloring** o coloración: es una asignación de etiquetas o colores de tal manera que dos vértices (nodos) de una misma arista (enlace) nunca tengan el mismo color.

Se trata de un algoritmo iterativo donde el primer paso es seleccionar los enlaces que se planificarán en el slot *k* en el que nos encontramos. Para ello se debe encontrar un matching en G, o lo que es lo mismo, seleccionar de entre todo el árbol los enlaces que construirán DCFL(k).

A continuación, se le debe asignar un channel offset a cada uno de estos enlaces. Con el fin de aprovechar al máximo tanto el espacio frecuencial como las ventajas ofrecidas por TSCH, se colocan en una misma celda todos aquellos enlaces seleccionados en el paso anterior que no interfieran entre sí. Esto se lleva a cabo construyendo un grafo l(k) y realizando vertex coloring sobre sus nodos.

Todo este procedimiento se ejecuta teniendo en cuenta los niveles de cola local y global de cada nodo, que se actualizan al final de cada iteración de acuerdo a la planificación que se ha hecho del slot.

Este tipo de algoritmos tiene las desventajas propias de los algoritmos centralizados, necesitando realizar un intercambio de toda la información de los nodos hasta un punto central para poder construir la planificación. Además, la planificación se realiza para unos valores iniciales dados, por lo que si se producen algunos cambios en la topología o en los valores de las colas es necesario volver a ejecutar el algoritmo y todo el proceso de intercambio de información.

2.1.3.3 DeTAS

El algoritmo Decentralized Traffic Aware Scheduling (DeTAS) [19] es un mecanismo de planificación distribuido que evoluciona de la versión centralizada de TASA. Se basa en intercambiar información sobre el tráfico que genera cada uno de los nodos para construir una planificación en función de las necesidades de cada nodo. Es capaz de construir una planificación óptima libre de colisiones. Aunque se defina como algoritmo distribuido, la información debe ser recogida por el coordinador para que construya la planificación y posteriormente la retransmita, pero tiene la diferencia de que los datos se van agregando con forme se transmite a su padre, de forma que la comunicación se realice entre pares de nodos adyacentes.

Para empezar, cada uno de los nodos debe conocer el tráfico que recibirá de sus hijos, así como el tráfico global que enviará a su padre, siendo este el trafico global de un nodo, formado por el tráfico del propio nodo más el de sus hijos. La información sobre el tráfico global de un nodo es la que se transmite de manera recursiva hasta llegar al coordinador, que calculará el número de slots totales necesarios para alojar todo el tráfico.

De esta forma se crean unas micro-planificaciones por cada uno de los grafos de encaminamiento, combinando cada una de estas micro-planificaciones para formar una macro-planificación. La planificación que construye el coordinador será común para todos los nodos, por lo que cada nodo no tomará ninguna decisión de planificación hasta que su padre indique los slots que le han sido asignados. Por lo tanto, el coordinador enviará la planificación a sus

hijos y se distribuirá en sentido descendente, indicando los slots que le han sido asignados hasta llegar a los nodos de mayor rango. En la Figura 12 (a) se puede ver una de las topologías en las que se utilizará el algoritmo DeTAS para construir su planificación, y en la Figura 12 (b) podemos ver como se distribuyen los slots entre los hijos.

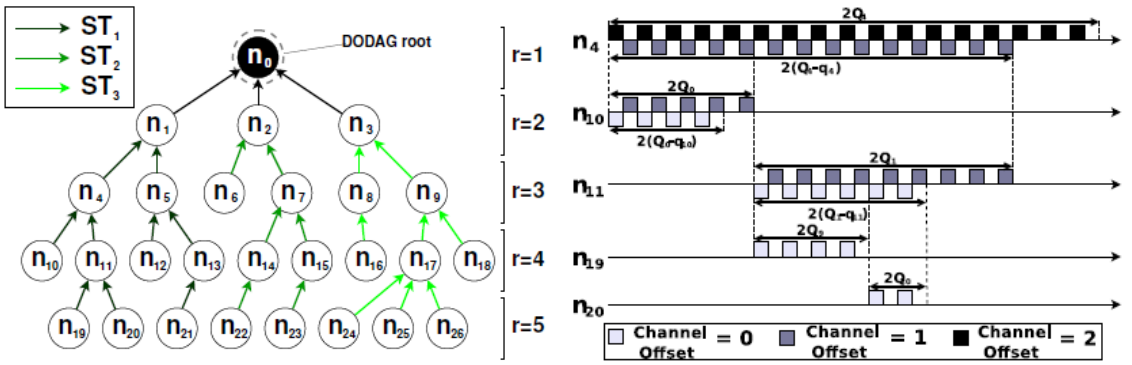


Figura 12. (a) Arquitectura en árbol DeTAS, (b) Planificación distribuida DeTAS

Este algoritmo toma además algunas otras consideraciones para evitar los solapamientos:

- Los slots de transmisión y de recepción están alternados temporalmente, por lo que se evita el riesgo de colisión.
- Se evita la contienda por los recursos temporales de los hijos de un nodo ya que sus slots irán consecutivos y sincronizados con los de su padre, consiguiendo así que dos nodos de un mismo rango no puedan enviar información en timeslots similares. Esto se puede ver en el ejemplo de la Figura 12 en el que los nodos n10 y n11 distribuyen sus slots en transmisión acordes con los de recepción de su padre n4 de manera consecutiva.
- El channel offset va cambiando con cada salto y no se reutiliza hasta haber realizado al menos tres saltos.
- Se definen planificaciones pares cuando sus celdas en transmisión comienzan en una posición par e impares cuando comienzan en posición impar. De esta manera se podrán combinar dos ramas diferentes sin que se solapen entre ellas.

Este algoritmo presenta algunos inconvenientes como el hecho de que la planificación que construye el coordinador se realiza para unas condiciones iniciales, por lo que, si el tráfico que generan los nodos varía, o se añaden o eliminan nodos a la red, la planificación ya no estará acorde con la nueva configuración. En la descripción del protocolo se exponen algunas consideraciones en las que es posible no volver a ejecutar todo el algoritmo si estas modificaciones no alteran la configuración del resto de la planificación.

2.1.3.4 Wave

Wave [20] es otro algoritmo de planificación distribuido también basado en el tráfico que genera cada uno de los nodos. Permite construir una planificación óptima libre de colisiones realizando la planificación en diferentes entapas denominadas olas (waves). El número total de iteraciones/olas dependerá del número máximo de paquetes a transmitir por uno de los nodos que forman la red.

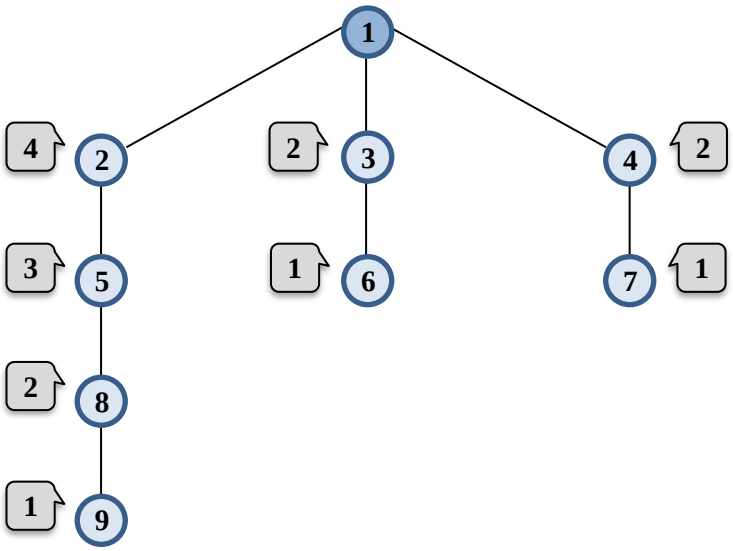
Cada uno de los nodos que forman la red deberá construir una lista con los nodos que haya dentro de su rango y con los que pueda entrar en conflicto. El proceso de planificación es iniciado por el coordinador de la red, que envía un mensaje de iniciación hacia todos los nodos. Los nodos que tengan un orden de prioridad superior serán los primeros en asignarse los recursos radio, este orden de prioridad dependerá del número de paquetes que tenga el nodo por transmitir, una vez tenga asignado estos recursos lo notificará a todos los nodos que estén en su lista de nodos en conflicto. De esta forma los nodos transmiten la información de su tráfico y de los slots y canales que sus nodos en conflicto han solicitado. Esta información se va transmitiendo en sentido ascendente hasta llegar al coordinador.

Una vez el coordinador ha recogido toda esta información envía una notificación que contiene la asignación de los slots de la primera ola junto con el factor de repetición, que dependerá de los paquetes que tenga que transmitir cada nodo. De esta forma en la primera ola todos los nodos vaciarán un paquete de su cola con la planificación que haya proporcionado el coordinador. En las siguientes olas, todos los nodos que hayan tenido asignado un slot en la primera ola le corresponderán transmitir en el mismo slot si dispone de paquetes que transmitir. De esta forma en cada ola se ira vaciando un paquete de las colas de todos los nodos, y aquellos que no tengan paquetes que transmitir desaparecerán de la planificación, reduciendo así el número de recursos que se utilizan en cada iteración.

En la Figura 13 se puede ver un ejemplo de una red formada por nueve nodos. El número de paquetes que tienen por transmitir en un instante de tiempo se indica en una etiqueta junto a cada nodo. Dado que el mayor número de paquetes a transmitir son los 4 paquetes del nodo 2 serán necesarias 4 olas para vaciar las colas de todos los nodos.

En la primera ola son necesarios 3 slots (representados en color azul en la tabla) para permitir que cada uno de los nodos envié su paquete sin entrar en colisión con el resto de nodos. En la segunda ola los nodos 6, 7 y 9 han vaciado su cola por lo que ya no entran en juego en la planificación, el resto de nodos cogen la misma planificación que en la primera ola. En esta segunda iteración también serán necesarios tres slots (representados en color verde).

En la tercera ola los nodos 3, 4 y 8 desaparecen de la planificación lo que permite reducir los slots asignados a 2 (representados en rojo). La cuarta y última ola el único nodo que tiene paquetes por transmitir es el nodo 2, por lo que solo será necesario un slot para que termine de vaciar su cola de paquetes.



slots / ch's	1	2	3	4	5	6	7	8	9
1	(2)(6)(7)	(3)(5)	(4)	(2)	(3)(5)	(4)	(2)	(5)	(2)
2	(8)	(9)		(8)					

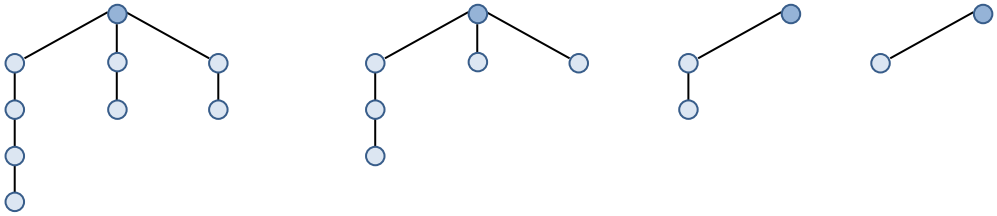


Figura 13. Ejemplo de planificación algoritmo WAVE.

Como inconvenientes tenemos problemas parecidos a los que ocurrían en DeTAS, si se producen cambios en el tráfico demandado por un nodo o aparecen nuevos nodos en la topología sería necesario volver a ejecutar el algoritmo, además de que una vez ejecutadas las olas pueden surgen cambios en el tráfico que no se tienen en cuenta durante la sucesión de las olas, ya que se basan en factores de repetición de la primera ola.

2.1.3.5 Orchestra

Orchestra [21] se define como mecanismo autónomo de planificación, ya que no se considera estrictamente como distribuido ya que no requiere de un intercambio adicional de información por parte de los nodos. Los dispositivos calculan su propia planificación a partir de la información que disponen sobre la topología y sus vecinos adyacentes. Se basa en información proporcionada por el protocolo RPL de encaminamiento y de la información que se intercambia mediante sus mensajes de control, por lo que evita enviar tráfico adicional.

En este mecanismo se emplean diferentes planificaciones que pueden ir actualizándose automáticamente e instantáneamente conforme la topología de la red va evolucionando. Por lo tanto, este mecanismo permite construir una planificación genérica y flexible utilizando para ello RPL, mientras se beneficia de la robustez del mecanismo de acceso al medio TSCH. Este mecanismo permite reducir las situaciones de contienda drásticamente o incluso eliminarlo en determinadas configuraciones.

Las planificaciones de Orchestra pueden contener diferentes slotframes de diferentes longitudes, pudiendo configurar cada uno de estos slotframes para que transmita diferentes tipos de tráfico como beacons de TSCH, señalización de control de RPL o mensajes de aplicación y datos. Estos slotframes tienen un periodo de repetición que son mutuamente primos entre ellos, asegurando que sus ciclos son independientes. En el caso de que varios slots de diferentes slotframes llegasen a solaparse, el slot que pertenezca al slotframe con una mayor prioridad será el que tenga preferencia. La clave en el funcionamiento autónomo de este mecanismo está en que, el nodo selecciona el slot temporal y el canal en el que transmitir en función de un identificador único basado en el receptor o el emisor que interviene en la comunicación. En la Figura 14 se puede ver una de estas planificaciones distribuidas en varios slotframes. En esta figura se puede apreciar cómo se planifican los slots temporales en el caso de que varios de ellos se solapen.

Figura 14. Distribución Slotframes Orchestra

Este ejemplo concreto mostrado en la figura contiene la siguiente planificación:

- Un slotframe que contiene slots dedicados para broadcast desde cada nodo hacia sus hijos para enviar las beacons de TSCH, con un periodo de repetición de 7 slots.
- Un slotframe con un slot común para todos los nodos de la red para enviar los mensajes de control, tanto broadcast como unicast, del protocolo RPL (DIO, DIS, DAO), con un periodo de repetición de 5 slots.
- Un slotframe que tiene configurados, un slot dedicado para comunicaciones unicast desde cada uno de los nodos hasta su padre, con un periodo de repetición de 9 slots. También se pueden configurar otros slots dedicados a comunicaciones unicast desde cada nodo hasta cada uno de sus hijos.

Además, Orchestra define diferentes tipos de slots dependiendo del tipo de comunicación para el que se esté utilizando. Los diferentes tipos se ilustran en la Figura 15 son los que se resumen a continuación.

- **Common Shared Orchestra Slots (CS)**. Estos slots consisten en un recurso compartido por todos los nodos de la red, utilizado tanto para transmitir como recibir información. Este slot se colocará en unas coordenadas fijas, lo que emula un comportamiento de enlace que siempre está disponible. Es importante recordar que el mecanismo TSCH utiliza un método de acceso aleatorio para resolver la contienda en este tipo de slots compartidos.
- **Receiver-based Shared Orchestra Slots (RBS)**. Estos slots se utilizan para establecer una comunicación entre dos vecinos, en un slot temporal y canal de comunicaciones que se derivan de las características del receptor. En cada nodo, un slot RBS implica que habrá planificado un slot para recibir y un slot para transmitir por cada uno de los

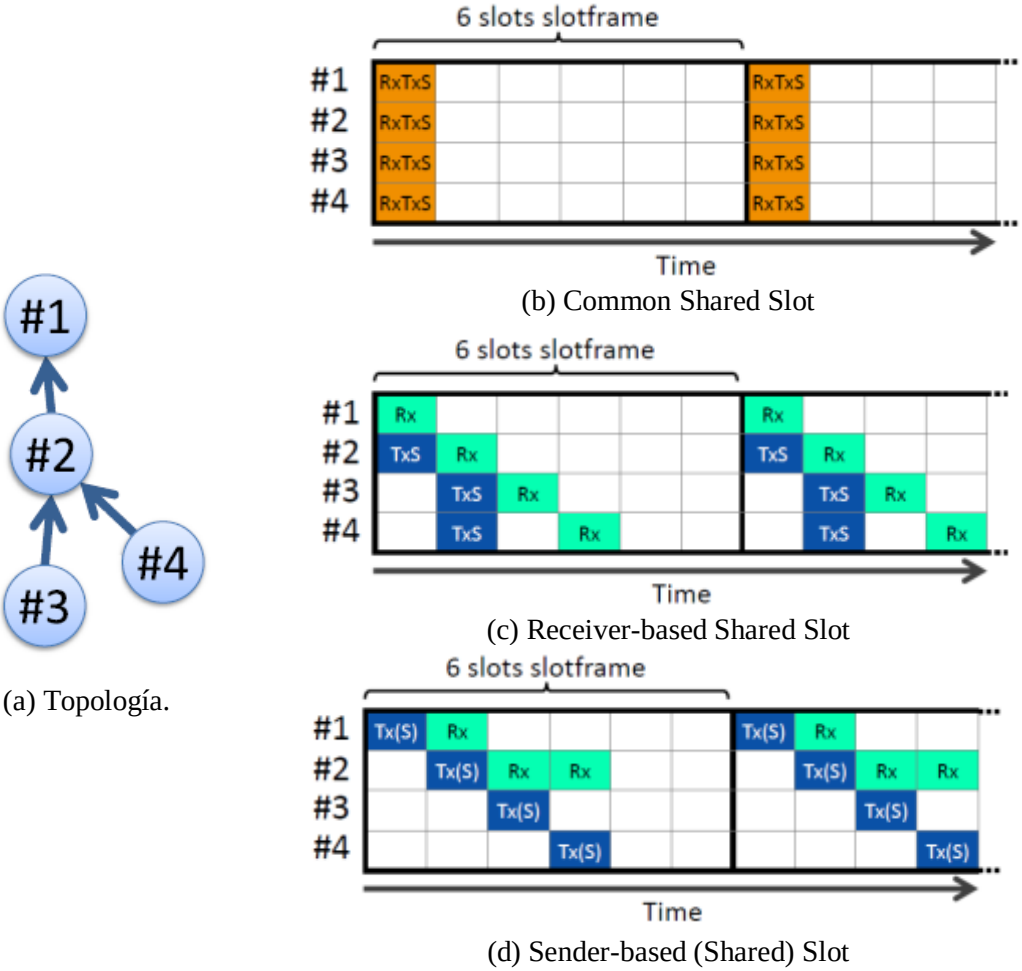
vecinos. Una forma de calcular las coordenadas (slot + channel offset) es utilizar una función hash de la dirección MAC del propio nodo, que es un identificador único dentro de la red.

Un ejemplo de utilización es la comunicación de hijo a padre en la que los padres están escuchando en un único slot para recibir cualquier tráfico procedente de sus hijos. En el caso de que un nodo cambie de padre, este actualizará su slot de transmisión de forma autónoma. Dado que varios hijos pueden intentar transmitir en el mismo slot hacia su padre, será necesario resolver esta situación mediante los mecanismos de contienda de TSCH.

- Sender-based Shared Orchestra Slots (SBS). Este tipo de slots son similares a los RBS, excepto que las coordenadas (slot y channel offset) se calculan a partir de las propiedades del emisor en lugar del receptor. En cada nodo, un slot SBS implica que habrá un slot para recibir por cada vecino y un único slot para transmitir. Esto provoca que los nodos tengan un consumo mayor de energía que para el caso de RBS (los slots para transmitir no tienen ningún coste si no hay tráfico, mientras que los slots para recibir requieren estar siempre activos esperando a que llegue algún paquete), sin embargo, ayuda a reducir la contienda.

  Un ejemplo sería, en una comunicación de hijo a padre, en la que los nodos tienen un slot para transmitir fijo y los padres mantienen un slot para recibir para cada uno de sus hijos. En el caso de que se produjera un cambio de padre, el hijo no necesita actualizar su slot para transmitir, pero el antiguo padre tendrá que eliminar el slot donde esperaba recibir información del antiguo hijo, y el nuevo padre deberá agregar un nuevo slot basado en las características del hijo.

- Sender-based Dedicated Orchestra Slots (SBD). En el caso de que podamos configurar un sloframe lo suficientemente largo para planificar un slot para transmitir para cada nodo, y suponiendo que todos ellos tienen un identificador único, la comunicación libre de contienda es posible. Orchestra permite realizar una planificación que esté libre de contienda utilizando los slots SBD, que son similares a los SBS excepto porque son recursos dedicados de TSCH.



(a) Topología.

(b) Common Shared Slot

(c) Receiver-based Shared Slot

(d) Sender-based (Shared) Slot

Figura 15. Ilustración de los diferentes tipos de slots en Orchestra en una red de 4 nodos.

Para mantener sus planificaciones Orchestra utiliza unas simples reglas de planificación. Este conjunto de reglas serán las que determinen los diferentes slotframes con los que contará la planificación global de la red. Cada una de estas reglas tendrá una serie de propiedades que definirán el slotframe y los slots que se planificarán en ellos. Este conjunto de propiedades son las que se muestran a continuación:

- ***Handle***. Es un número entero positivo que sirve tanto para identificar como asignar una prioridad al slotframe. Un número menor indica una prioridad más alta. Este parámetro forma parte de las características definidas en TSCH.
- ***Length***. Indica el número de slots dentro de un slotframe. Deberá ser mutuamente primo con las longitudes del resto de slotframes de la red.
- ***Traffic Filter***. Sirve para identificar los diferentes planos de tráfico que circulan por la red, para ser capaces de mapearlos en múltiples slotframes.
- ***Neighbors***. Indica el vecino o conjunto de vecinos que van a utilizar el slot, como el padre directo o el conjunto de hijos.
- ***Coordenadas***. Indican el slot temporal y el channel offset dentro de un slotframe e identifican un enlace de comunicaciones.
- ***Options***. Se trata de las opciones estándar de TSCH, como slot para recibir (RX), transmitir (TX) o compartidos (S).

2.1.4 Protocolo 6LoWPAN

El estándar 6LowPAN conocido como *IPv6 over Low Power Wireless Personal Area Network* fue desarrollado por la IETF con el objetivo era encontrar una solución al transporte de paquetes IPv6 sobre tramas IEEE 802.15.4. Sus características vienen especificadas en diferentes RFCs [22] [23] [24].

El estándar IEEE 802.15.4 especifica un tamaño de paquetes MTU de 127 bytes, dejando unos 80 octetos para la carga útil de capa MAC si tuviésemos características de seguridad habilitadas, en unos enlaces inalámbricos con un throughput de 250 kbps o menos. El formato de adaptación que plantea 6LoWPAN permite transportar datagramas de IPv6 a través de estos enlaces tan limitados, teniendo en cuenta sus limitaciones de ancho de banda, memoria o recursos energéticos. Por lo tanto, este protocolo se encarga de definir los mecanismos de compresión de cabeceras necesarios para hacer que IPv6 resulte práctico en redes IEEE 802.15.4.

El tamaño de la unidad máxima de transmisión (MTU) de los paquetes en IPv6 es de 1280 octetos, por lo que resulta imposible transmitir esta carga de datos a través de redes IEEE 802.15.4 ya que su tamaño máximo es de 127 bytes. Sin embargo, este tipo de redes no están pensadas para que utilicen todas las características que implementa IPv6, por lo que muchas cabeceras se podrían omitir, además de que el tamaño de los mensajes que se transmiten en estas redes no suele ser muy grande. Por lo tanto, teniendo en cuenta la pequeña carga de datos de estas aplicaciones junto a la apropiada compresión de las cabeceras de IPv6 se podrían ajustar estos paquetes para que se transmitieran a través de una única trama de IEEE 802.15.4, sin necesidad de llegar a fragmentar el paquete.

Esta compresión y utilización de arquitecturas IP en redes inalámbricas de are personal presentan importantes ventajas como las que se muestran a continuación:

- La utilización de un direccionamiento IP permite utilizar una infraestructura de red ya existente, permitiendo que redes que implementen el estándar IEEE 802.15.4 puedan comunicarse más fácilmente con redes externas basadas en IP.
- Se trata de una infraestructura ya madura, por lo que existen multitud de herramientas que permiten diagnosticar y gestionar redes IP.
- Permite utilizar tanto direcciones MAC cortas de 16 bits como las extendidas de 64 bits, lo que se ajusta a la idea de desplegar grandes redes de pequeños dispositivos inalámbricos.

2.1.4.1 Pila de protocolos

La Figura 16 muestra la representación de la pila de protocolos donde se incluiría la capa de adaptación de 6LoWPAN.

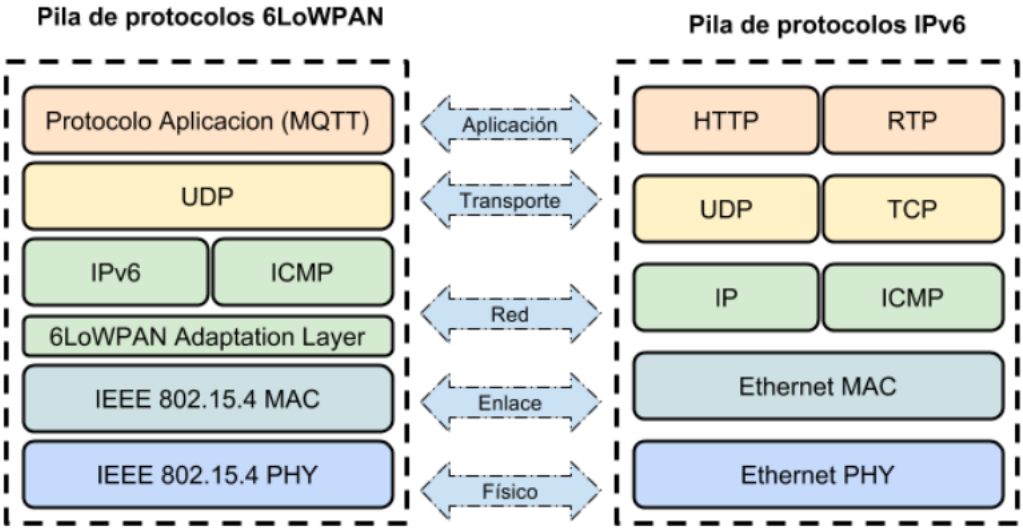


Figura 16. Pila de protocolos 6LoWPAN frente IPv6

Como se puede ver, este subnivel formaría parte de la capa de red, sirviendo como capa de adaptación entre la capa MAC de IEEE 802.15.4 y la capa de red de IPv6. En la capa de transporte vendría representado el protocolo UDP, el cual puede ser comprimido mediante el protocolo 6LoWPAN ya que en este también se contempla. Por otro lado, el protocolo TCP no se contempla en la utilización de este tipo de redes por motivos de rendimiento y complejidad que exceden las capacidades de estos pequeños dispositivos.

2.1.4.2 Formato de datagramas 6LoWPAN

La Figura 17 muestra el formato de un paquete 6LoWPAN, como ya hemos comentado, este no podrá superar los 127 bytes.

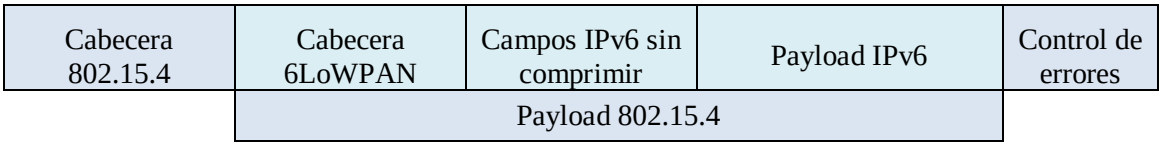


Figura 17. Formato de mensaje 6LoWPAN.

Todas las cabeceras de adaptación comenzarán con un campo de 8 bits, conocido como dispatch byte y que permitirá identificar el tipo de cabecera que sigue a continuación. En la Figura 18 se muestran los posibles valores que puede tomar el campo Dispatch.

00	No es 6LoWPAN
01	Cabecera Comprimida
10	Cabecera Mesh
11	Cabecera Fragmentación

00	xxxxxx	No es 6LoWPAN
01	000001	IPv6 sin compresión
01	000010	IPv6 con compresión HC1
01	010000	Broadcast BC0
01	1xxxxx	IPv6 con compresión IPHC
01	111111	Código escape para alargar dispatch
10	xxxxxx	Enrutamiento Mesh
11	000xxx	Fragmentación (primer fragmento)
11	100xxx	Fragmentación (continuación)

Figura 18. Posibles valores del campo Dispatch.

2.1.4.3 Compresión de cabeceras IPv6

El formato de adaptación de 6LoWPAN fue especificado en una primera RFC4944 en la que se definía los mecanismos de retransmisión, fragmentación de cabeceras y la compresión de datagramas IPv6 mediante las cabeceras HC1 (Header Compression 1) y HC2 (Header Compression 2) que permitían reducir la longitud de las cabeceras tanto de IPv6 como del protocolo UDP. Sin embargo, estos dos métodos de compresión resultaban insuficientes para la mayoría de usos prácticos de IPv6 en redes 6LoWPAN. HC1 era más efectivo para comunicaciones link-local unicast en la que se transmitía el prefijo de la dirección de enlace-local y un identificador de interfaz obtenidos directamente de las direcciones de IEEE 802.15.4. Aunque esto resulta de utilidad en el descubrimiento de vecinos IPv6, para el protocolo DHCPv6 o para protocolos de enrutamiento, normalmente no tienen ninguna utilidad para tráfico de datos a nivel de aplicación, por lo que este mecanismo de compresión es bastante limitado.

Debido a esto se volvieron a definir otros mecanismos de compresión que sustituyeran a los antiguos en una nueva RFC6282 en el que se detallan los mecanismos de compresión IPHC (IP Header Compression) para comprimir eficientemente las direcciones IPv6 Local Única, Global y multicast, que incluyen algunas mejoras al formato definido por la antigua RFC. Además, se define otro mecanismo de codificación NHC (Next Header Compression) de cabeceras arbitrarias que puede utilizarse para comprimir datagramas UDP.

A continuación, se detallan estos métodos que acabamos de comentar, tanto los nuevos mecanismos descritos en la RFC6282 como los HC1 y HC2 ya que se pueden utilizar para descomprimir paquetes basados en la versión obsoleta.

2.1.4.3.1 Compresión HC1-HC2

Partiendo del hecho de que los nodos se van a unir a la misma red 6LoWPAN, estos dispositivos compartirán algunos estados lo que hace posible comprimir algunas cabeceras que contengan información explicita que no sea relevante para establecer los enlaces de comunicación entre varios dispositivos. Las siguientes cabeceras IPv6 son algunos de los valores que serán comunes en redes 6LoWPAN, por lo que la cabecera HC1 se construye eficientemente para que los comprima:

- La versión será siempre IPv6.
- Tanto la dirección de origen como de destino serán de enlace local.
- Los identificadores de interfaz se pueden deducir de las direcciones de origen y destino que se obtengan de la capa MAC.
- La longitud del paquete también la podemos obtener del nivel MAC.
- Los valores de Traffic Class y de Flow Label serán cero.
- La siguiente cabecera siempre será UDP, ICMP o TCP.

El único campo que siempre necesita transmitirse completamente es el de Límite de Saltos (8 bits). Esta cabecera IPv6 de 40 octetos se puede llegar a comprimir hasta dejarla únicamente en 2 octetos (1 para la codificación HC1 y 1 octeto para el Límite de Saltos que se enviará integro).

Como puede verse en la Figura 19, el formato comienza utilizando un valor de Dispatch que indica que le sigue la cabecera de compresión HC1. Además, uno de los bits que forma parte de esta cabecera indica si la siguiente cabecera HC2 se incluye o no. En el caso de que se utilice el protocolo UDP esta cabecera HC2 deberá de incluirse para comprimir los datos de la cabecera de UDP.

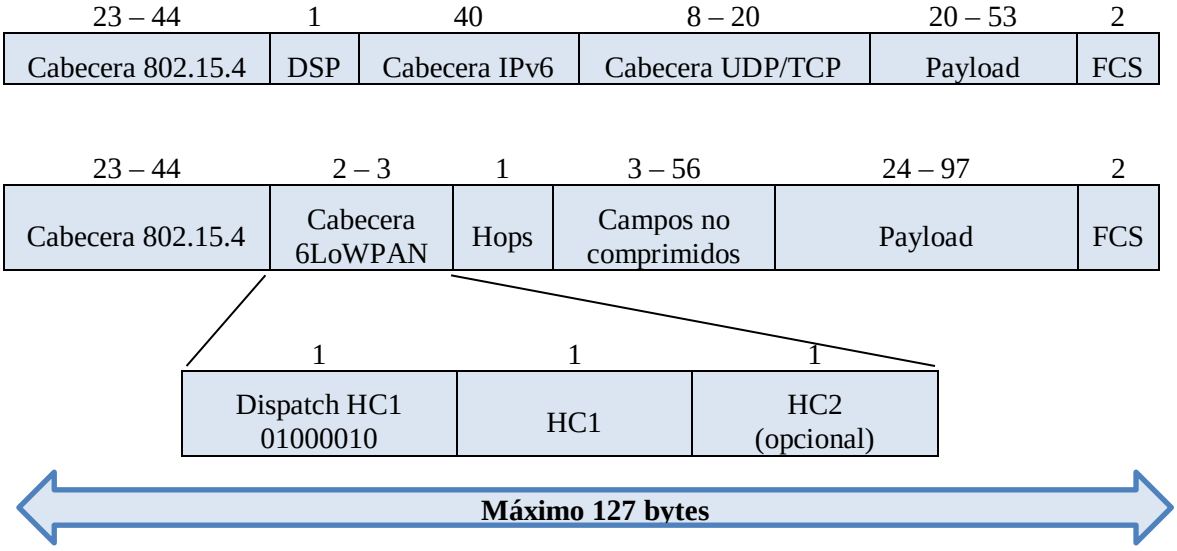


Figura 19. Compresión de cabeceras HC1-HC2.

2.1.4.3.2 Compresión IPHC

Esta codificación es la que se encuentra descrita en la RFC6282, donde se recomienda no utilizar las anteriores cabeceras HC1 y HC2 aunque existe una interoperabilidad entre ellas. La codificación IPHC utiliza 13 bits, de los cuales se utilizan los 5 bits más significativos del campo de tipo de dispatch. Además, la codificación se puede extender con otro octeto para permitir añadir contenido adicional. Todos aquellos campos de cabecera que se incluyan directamente en el paquete deberán ir seguida a la codificación IPHC. En el mejor de los casos, la codificación IPHC puede comprimir las cabeceras de IPv6 hasta dejarlas en dos octetos en el caso de comunicaciones de enlace local. Cuando se encamina a través de múltiples saltos, IPHC puede comprimir las cabeceras de IPv6 hasta dejarlo en 7 octetos (1 para el dispatch, 1 para la cabecera IPHC, 1 para el límite de saltos, 2 octetos para la dirección de origen y 2 octetos para la dirección de destino).

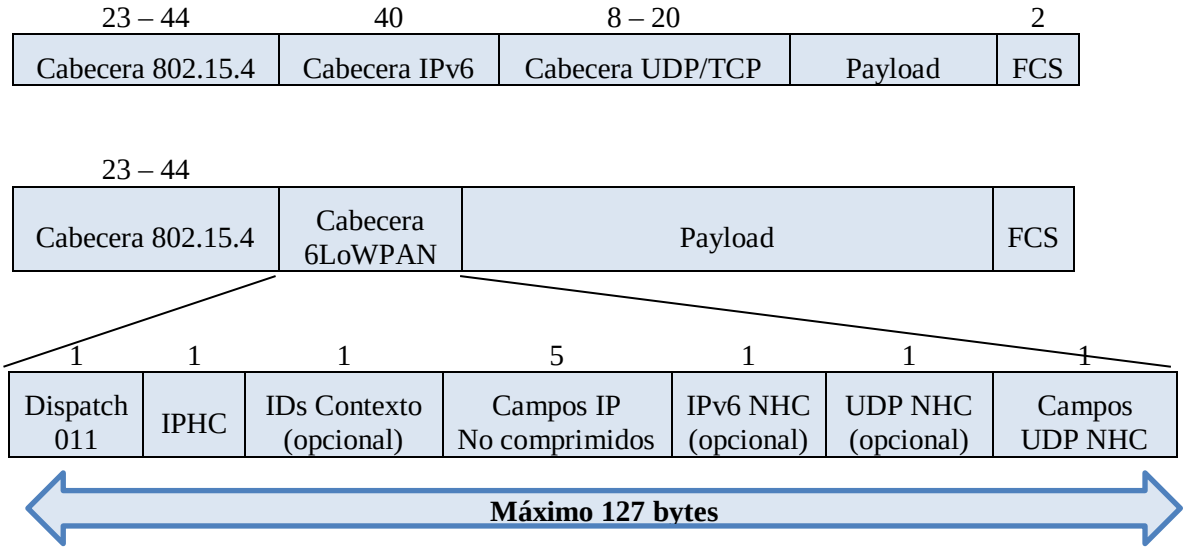


Figura 20. Compresión de cabeceras IPHC-NHC.

El formato de compresión para diferentes cabeceras viene identificado en uno de los campos inmediatamente después de la cabecera comprimida de IPHC. Estas cabeceras irán en uno de los campos de NHC, pudiendo incluir cualquier cabecera añadida de IPv6, UDP u otro protocolo similar.

2.1.5 Routing Protocol for Low-Power and Lossy Networks (RPL)

Las Low-Power and Lossy Networks (LLNs) son un tipo de redes en las que tanto los dispositivos enrutadores como aquellos que están interconectados a través de la red están limitados, en potencia de procesamiento, memoria y energía (suministrada normalmente con baterías). Sus enlaces están caracterizados por tener unas altas tasas de pérdidas, baja velocidad e inestabilidad del canal radio. Este tipo de redes permiten formar diferentes estructuras y soportan diferentes tipos de tráfico como punto-a-punto (conexiones entre dispositivos dentro de la LLN), punto-a-multipunto (tráfico desde un punto central hacia un conjunto de nodos destino) y multipunto-a-punto (tráfico desde los dispositivos de la LLN hacia un nodo central).

RPL [25] es un protocolo de encaminamiento dinámico especialmente diseñado para las WSN. Fue desarrollado por el grupo IETF ROLL como mecanismo de enrutamiento dinámico del tipo vector distancia, para conseguir cumplir con los requisitos que el propio grupo de trabajo a definido en los documentos [26] [27] [28].

En una red es posible estar ejecutando múltiples instancias de RPL de forma conjunta, pudiendo tener cada una de ellas diferentes criterios de funcionamiento.

2.1.5.1 Topologías en RPL

En este apartado vamos a describir los aspectos básicos que tienen las topologías construidas por RPL, y aquellas reglas que se siguen para construirlas. Este tipo de topologías vienen representadas por lo que se conoce DAG (Directed Acyclic Graph) que no es más que un grafo en el que todos los enlaces están organizados de tal forma que no se produzcan ciclos. Las rutas de RPL están optimizadas para un tráfico que lleva hasta uno o varios nodos raíces (roots nodes) que actuarán como sumidero de la topología. Por lo tanto, RPL organiza la topología como un grafo DAG que es dividido en uno o más DODAG (Destination-Oriented DAG), uno por cada sumidero. Para poder identificar y mantener esta topología RPL utiliza los siguientes valores:

- **RPLIntanceID**. Se trata de un identificador único que agrupa a uno o varios DODAGs. Una red puede tener varios RPLInstanceIDs, en el que cada uno identificará a un conjunto independiente de DODAGs que podrán ser optimizadas de manera diferente o tratadas para diferentes aplicaciones. Todos los DODAG que pertenezcan a la misma instancia de RPL utilizarán la misma Función Objetivo (OF) para el cálculo de sus rutas.
- **DODAGID**. La combinación de RPLInstanceID y DODAGID identifica únicamente a un único DODAG en la red. Una instancia RPL puede tener múltiples DODAGs, en la que cada una de ellas tiene un único DODAGID.
- **DODAGVersionNumber**. Un DODAG puede ser reconstruido por un DODAG root, momento en el cual incrementará el valor de DODAGVersionNumber.
- **Rank**. El Rank o rango establece un orden parcial dentro de un DODAG, definiendo la posición de un nodo de manera individual respecto a la posición del DODAG root.

RPL proporciona las rutas ascendentes desde cada uno de los nodos de la red hasta el DODAG root, formando un DODAG optimizado de acuerdo a una Función Objetivo (OF). La formación y mantenimiento de este DODAG se realiza a través de unos mensajes llamados DIO (DODAG Information Object). Las funciones objetivo definen como los nodos de RPL deben seleccionar y optimizar las rutas dentro de una RPLInstance. En estas definiciones se encuentran como los nodos transforman una o varias métricas en un determinado rango, que es la distancia aproximada hasta el DODAG root. Algunas de estas métricas vienen definidas en [29] y permiten a un nodo seleccionar cuál de sus vecinos es más apropiado para ser su padre.

La construcción del DODAG sigue el siguiente procedimiento:

1. Uno o varios nodos de la red serán los encargados de actuar como DODAG roots, y proporcionarán la configuración asociada al DODAG al resto de nodos que la compartan.
2. Los DODAG root comienzan a advertir de su presencia difundiendo mensajes DIO a todos los nodos RPL, con información relevante como la pertenencia a un determinado DODAG, coste de encaminamiento que estará relacionado con el rango, así como otras métricas relevantes para la formación del DODAG.
3. Los nodos estarán escuchando a la espera de recibir los mensajes DIOs y utilizarán su información para unirse a un nuevo DODAG (seleccionando así al que será su padre) o también para mantener y actualizar la información de un DODAG ya existente, de acuerdo a las normas especificadas en la Función Objetivo y al rango de sus vecinos. Los nodos que estén recibiendo los mensajes DIOs mantienen una lista de vecinos que pueden ser padres potenciales, escogiendo a aquellos que tengan un rango inferior al suyo, ya que estarán más cerca del DODAG root y teniendo en cuenta también las métricas definidas por la Función Objetivo.
4. Con forme los nodos van recibiendo mensajes DIO de sus vecinos, van construyendo diferentes entradas en una tabla de encaminamiento para cada dirección especificada en los mensajes DIO. Una vez escogido cuál de estos vecinos será su padre, el nodo calcula el valor de su Rank para después volver a propagar el mensaje DIO para que los nodos que haya por debajo puedan realizar el mismo proceso.

Este procedimiento se puede ver representado en la Figura 21.

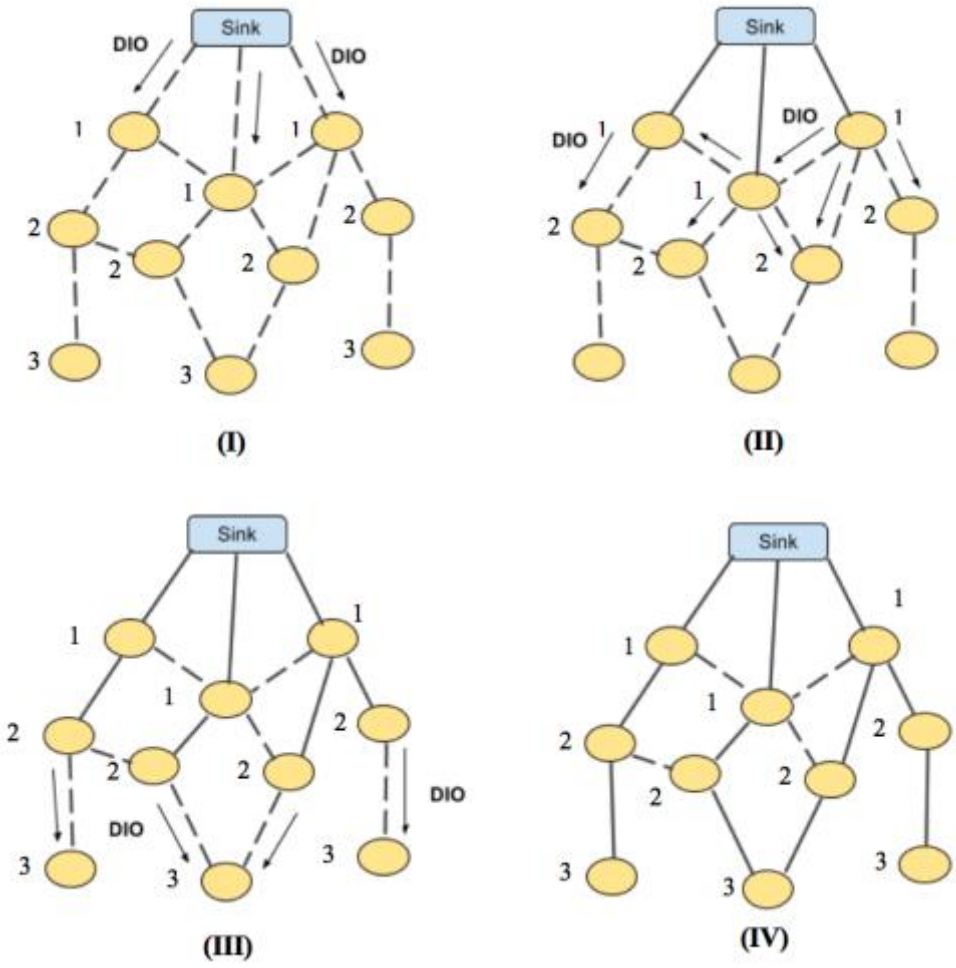


Figura 21. Formación del DODAG en RPL.

2.1.5.2 Flujo de tráfico RPL

Por otro lado, la formación de rutas descendentes se lleva a cabo utilizando unos mensajes llamados DAO (Destination Advertisement Object) que permiten establecer este tipo de rutas. Este tipo de mensajes son una característica opcional para aplicaciones que requieran de comunicaciones punto-a-multipunto (P2MP) o punto-a-punto (P2P). El protocolo RPL soporta dos tipos de tráfico descendente: Storing y Non-Storing. Cualquier instancia de RPL puede utilizar tráfico con el Storing mode o bien con el Non-Storing mode. En el Non-Storing mode los paquetes se transmiten hasta llegar hasta el DODAG root antes de que este lo vuelva a retransmitir hacia su destino. En el caso del Storing mode, el paquete puede ir directamente hasta su destino si se encuentra en una ruta descendente del propio nodo o de alguno de sus ancestros antes de alcanzar al DODAG root.

2.1.5.3 Mensajes de control ICMPv6 en RPL

El protocolo RPL emplea el protocolo ICMPv6 para intercambiar los mensajes de control necesarios para la formación y mantenimiento del DODAG. Estos mensajes de control de RPL vienen identificado por un código que indicará que tipo de mensaje es y que opciones incluye.

La mayoría de mensajes de Control en RPL suelen tener el alcance de un enlace. La única excepción son los mensajes DAO / DAO-ACK que se intercambian en el Non-Storing mode, los cuales se intercambian utilizando una dirección unicast a través de múltiples saltos, por lo que deben utilizar direcciones globales lo local únicas tanto para la dirección de origen como la de destino. Para el resto de mensajes de control RPL, la dirección de origen es una dirección de enlace local y la dirección de destino es una dirección multicast dirigida a todos los nodos RPL o la dirección de enlace local unicast del destino. La dirección que representa a todos los nodos RPL viene definida en el protocolo ICMPv6 [RFC4443] y tiene un valor de ff02::1a.

La estructura de los mensajes de control RPL se compone de una cabecera ICMPv6 seguido del cuerpo del mensaje, que contendrá un campo base del mensaje que dependerá del tipo y un posible número de opciones. En la Figura 22 se puede ver una representación de un mensaje de control en RPL y en la Tabla 2 se recogen los posibles valores que puede tomar el código que identifica al tipo de mensaje de control.

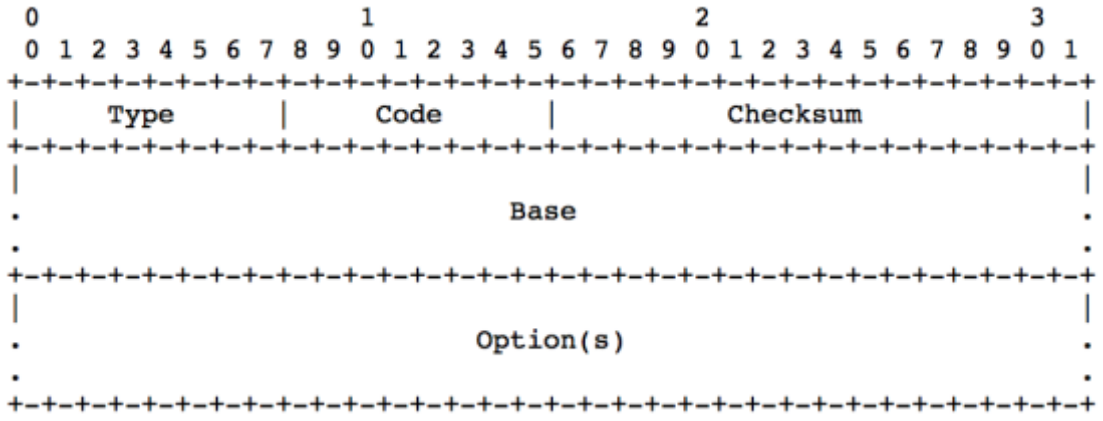


Figura 22. Estructura tipo de los mensajes de control de RPL

0x00	DODAG Information Solicitation (DIS)
0x01	DODAG Information Object (DIO)
0x02	Destination Advertisement Object (DAO)
0x03	Destination Advertisement Object Acknowledgment
0x80	Secure DIS
0x81	Secure DIO
0x82	Secure DAO
0x83	Secure DAO-ACK
0x8A	Consistency Check

Tabla 2. Posibles valores del campo Code de los mensajes de control RPL.

2.1.5.3.1 DODAG Information Solicitation (DIS)

Este tipo de mensajes de control puede ser utilizado por cualquier nodo RPL para solicitar que se le envíe un mensaje DIO que contenga la información necesaria para identificar e incluirse en un DODAG. Este tipo de mensaje no suele contener información relevante, por lo que el mensaje estaría compuesto por los siguientes dos campos: un octeto de Flags, que se inicializan a cero por el emisor y son ignoradas por el receptor, y otro campo de 8 bits sin utilizar reservado para posibles actualizaciones del protocolo. Como el resto de mensajes de control, puede ir seguido de una serie de opciones si se quisieran incluir, pero dado que este mensaje solo se utiliza para solicitar un DIO no se suelen utilizar estos campos.

2.1.5.3.2 DODAG Information Object (DIO)

Estos mensajes de control son quizá los más relevantes, ya que permiten crear la estructura del DODAG, así como mantener esta información actualizada mediante el intercambio entre los vecinos de la red. Además, nos servirán para realizar algunas implementaciones que desarrollaremos en el Capítulo 4 de esta memoria. El formato de estos mensajes es el que se muestra en la Figura 23.

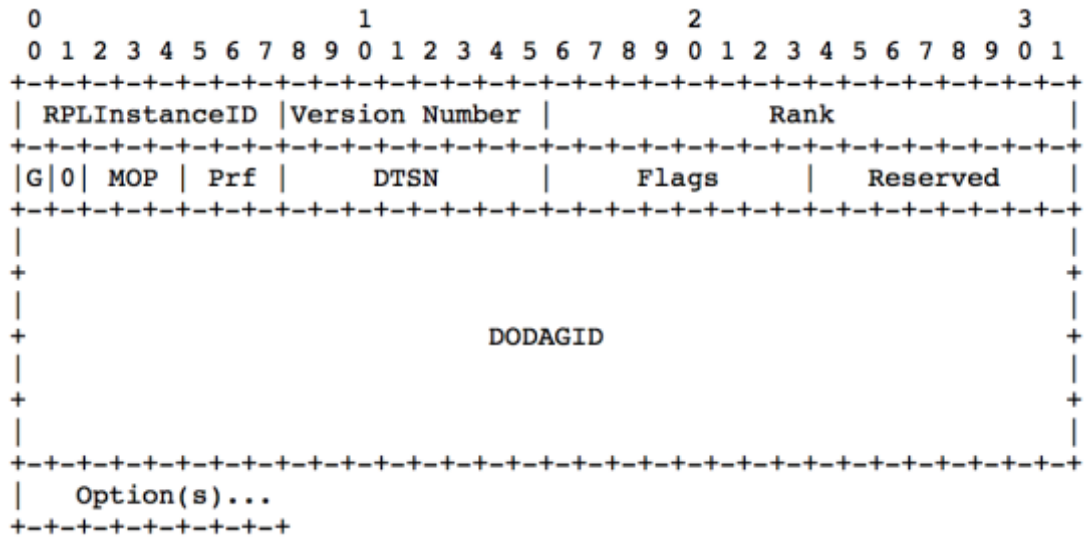


Figura 23. Formato de los mensajes DIO de RPL.

A continuación, se detalla el propósito de los diferentes campos de los mensajes DIO.

- Grounded (G): se trata de un flag que indica si el DODAG puede satisfacer los objetivos definidos por la aplicación en RPL.
- Mode of Operation (MOP): Este campo identifica el modo de operación de la Instancia RPL que viene configurado por DODAG root. Los nodos que se unan al DODAG deben cumplir con lo indicado en el campo MOP para poder participar como un router, o bien como un nodo hoja. Los diferentes modos de operación que pueden ir indicados en este campo son los que se muestran en la Figura 24.

MOP	Description
0	No Downward routes maintained by RPL
1	Non-Storing Mode of Operation
2	Storing Mode of Operation with no multicast support
3	Storing Mode of Operation with multicast support
	All other values are unassigned

Figura 24. Codificación de los modos de operación (MOP)

- DODAGPreference (Prf): Se trata de un número entero de 3 bits que define el orden de preferencia del DODAG root para poder compararlo con otros DODAG roots dentro de la misma instancia. Sus valores van desde 0x00 para el menos preferido hasta 0x07 para el más preferido.
- Version Number: Es un valor de 8 bits configurado por el DODAG root y que representa la versión del DODAG.
- Rank: Es un entero sin signo de 16 bit que representa el rango del nodo que ha enviado el mensaje DIO. De esta forma el receptor puede conocer el rango de todos los vecinos de los que haya recibido mensajes de control, y así realizar el proceso de selección de padre utilizando la función objetivo.
- RPLInstanceID: Campo de 8 bits configurado por el DODAG root que indica cual es el valor de la instancia RPL de la que forma parte el DODAG.
- Destination Advertisement Trigger Sequence Number (DTSN): Es un valor de 8 bits configurado por el nodo emisor del mensaje DIO. Este parámetro se utiliza como parte del proceso de mantenimiento de rutas descendentes.
- Flags: Este campo está reservado para ser utilizado por diferentes flags aún no implementados. Cuando se envía un DIO el emisor de inicializar a cero este campo y el receptor deberá ignorarlo.
- Reserved: Otro campo de 8 bits reservado para futuras características.
- DODAGID: Dirección Ipv6 de 128 bits configurada por el DODAG root que identifica inequívocamente a un DODAG. Esta debe ser una dirección Ipv6 enrutable perteneciente al DODAG root.

La transmisión de los mensajes DIO se realiza utilizando un Trickle timer que permite reducir el tráfico de control que circula por la red cuando esta permanece estable y sin cambios. Cuando se recibe un DIO de un emisor con un Rank menor y que no pueda causar cambios a su conjunto de posibles padres, su padre preferido o al propio rango podemos considerar este evento como consistente respecto al Trickle timer.

Por el contrario, los siguientes paquetes o eventos se considerarán como inconsistentes, por lo que causarán que el Trickle timer se reinicie:

- Cuando un nodo detecta una inconsistencia al reenviar un paquete.
- Cuando un nodo recibe un mensaje DIS multicast sin la opción de información solicitada, a menos que el mensaje DIS tenga habilitado el flag que restringe esta característica.
- Cuando un nodo recibe un mensaje DIS multicast con la opción de información solicitada y el nodo encuentra todas las opciones en el campo de opciones de información solicitada, a menos que el mensaje DIS tenga habilitado el flag que restringe esta caracteristca.
- Cuando un nodo se une a una nueva versión del DODAG (por ejemplo, actualizando su valor del DODAGVersionNumber, uniéndose a una nueva instancia de RPL, etc.).

Esta lista muestra solo algunos casos, y en la especificación se indica que se deberán considerar otro tipo de menajes y eventos que puedan ser inconsistentes con el actual DODAG.

Los parámetros utilizados para configurar este Trickle timer son los siguientes:

- Imin: se extrae de los mensajes DIO como (2^DIOIntervalMin) ms.
- Imax: se extrae delos mensajes DIO como DIOIntervalDoublings.
- k: se extrae de los mensajes DIO como DIORedundancyConstant. En RPL, cuando k toma un valor de 0x00, el valor debe ser tratado como una constante de redundancia igual a infinito, lo que indicaría que el Trickle timer nunca suprimirá los mensajes.

2.1.5.3.3 Destination Advertisement Object (DAO)

Estos mensajes de control se utilizan para propagar información de destino en rutas ascendentes a lo largo del DODAG. En el Storing mode, los mensajes DAO se transmiten entre el hijo y el padre o padres seleccionados. En el Non-Storing mode, los mensajes DAO son mensajes unicast dirigidos hacia el DODAG root. Estos mensajes opcionalmente pueden reenviar un DAO-ACK de regreso para comprobar la recepción del DAO por el emisor. El formato de los mensajes DAO es el que se muestra en la Figura 25.

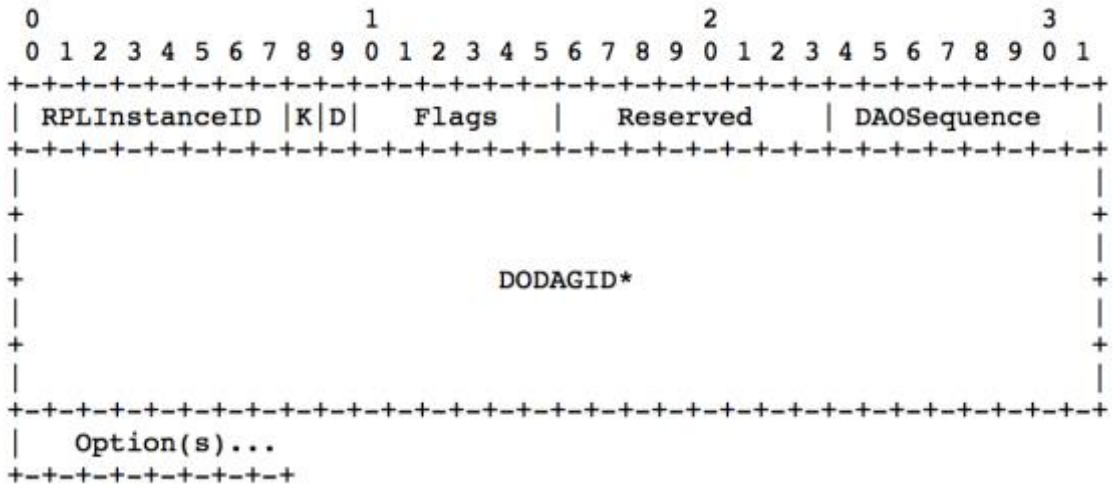


Figura 25. Formato del objeto DAO en RPL.

El propósito de los diferentes campos que forman el objeto DAO se detallan a continuación:

- K: El flag 'K' indica que el receptor debe transmitir un DAO-ACK de vuelta para confirmar la recepción del mensaje DAO.
- D: El flag 'D' indica que el campo DODAGID está presente en el mensaje, ya que puede omitirse. Este parámetro se debe configurar cuando se esté utilizando una instancia RPL local.
- Flags: Los 6 bits restantes no se utilizan y están reservados para poder utilizarse como flags. El campo se deberá inicializar a cero por el emisor y deberá obviarse por el receptor.
- Reserved: Se trata de 8 bits que están reservados para futuras características del protocolo. El campo se deberá inicializar a cero por el emisor y deberá obviarse por el receptor.
- DAOSequence: Es un valor que se incrementa por cada mensaje DAO único desde un nodo y repetido en el mensaje DAO-ACK. Permite por tanto identificar los mensajes DAO y emparejarlos con su correspondiente DAO-ACK.
- DODAGID (opcional): Es un entero sin signo de 128 bits configurado por el DODAG root que identifica a un único DODAG. Este campo solo está presente cuando el flag 'D' esté activado.

2.1.5.3.4 DAG Metric Container

Los DAG Metric Container son una de las opciones interesantes que pueden incluirse en los mensajes de control DIO o DAO. El formato que tienen es el que se muestra en la Figura 26.

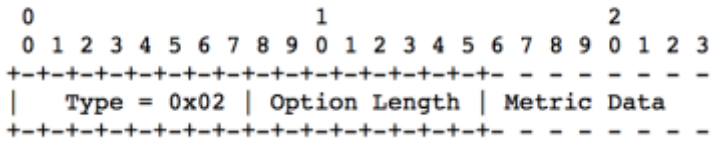


Figura 26. Formato de los DAG Metric Container

Esta opción puede ser utilizada como contenedor para transmitir cualquier tipo de métrica a lo largo del DODAG. Estos campos pueden contener una métrica que represente un número discreto de nodos, las características referentes a la calidad de los enlaces, ya sea de manera individual o de forma agregada a lo largo de todo el camino hasta el DODAG root. Esta mecánica resulta muy interesante para ser capaces de transmitir cualquier tipo de métrica que nos interese implementar para poder ser utilizada por la Función Objetivo para el cálculo del rango y la selección de padres.

Los DAG Metric Container pueden aparecer más de una vez en el mismo mensaje de control RPL, pudiendo utilizarse para transmitir varias métricas para poder utilizar una combinación de varias de ellas. En la especificación [RFC6551] se recogen algunas de las métricas planteadas por el grupo IETF para realizar el cálculo del rango y seleccionar entre los diferentes caminos a través de las funciones objetivo de RPL. Además, se describe en mayor profundidad la utilización de los Metric Container y de los diferentes tipos de métricas con los que se puede utilizar.

2.2 Selección de plataformas

2.2.1 Contiki OS

Contiki [30] [31] es un sistema operativo de código abierto pensado para el Internet de las Cosas, consiguiendo conectar pequeños microcontroladores de bajo coste y potencia a internet mediante un conjunto de herramientas que permiten construir sistemas inalámbricos complejos. Está pensado para implementar sistemas de comunicaciones de baja potencia como son las WSN, utilizando para ello diferentes estándares inalámbricos como 6LoWPAN, RPL o CoAP entre otros muchos. Este sistema operativo se ha desarrollado por un equipo de desarrolladores de todo el mundo y tiene importantes contribuidores como Atmel, Cisco, ETH, Redwire, Thinsquare y muchos otros.

Este sistema permite implementar una pila completa para redes IP, con algunos de sus protocolos como UDP, TCP y HTTP además de nuevos estándares de baja potencia destinados a las WSN. Está diseñado para poder operar en sistemas de muy baja potencia como algunos dispositivos que necesitan estar activos durante largos periodos de tiempo con baterías muy limitadas en cuanto a la energía que pueden ofrecer. Ya que está pensado para que pueda operar en este tipo de dispositivos y poder desarrollar sistemas eficientes energéticamente, Contiki proporciona mecanismos que permiten estimar la potencia consumida de los sistemas para comprender en que se está consumiendo esta energía.

Los requerimientos de memoria que utiliza Contiki no suelen necesitar de más de 10 k de memoria RAM y de 30 k de ROM, por lo que no supone problema en implementarlos en estos dispositivos cuyas capacidades de memoria son bastante limitadas. Para conseguir ahorrar memoria, pero proporcionando un buen control del flujo en el código, Contiki utiliza un mecanismo llamado protothreads, que permite combinar las llamadas a eventos con una programación multi-hilo lo que permite combinar diferentes bloques de código que se ejecuten cuando un evento ocurra.

El código de este sistema está desarrollado en C, y funciona en una amplia variedad de plataformas hardware, y tiene una comunidad consolidada de desarrolladores que apoyan la plataforma. Entre las plataformas hardware que son compatibles con Contiki tenemos una amplia lista [32] entre los que destacan algunos como:

- CC2538, un poderoso microcontrolador SoC para aplicaciones IEEE 802.15.4 en 2.4GHz compatible con 6LoWPAN y ZigBee.
- SensorTag, un sistema de prototipado para dispositivos IoT. Bastante interesante dado su pequeño tamaño y su capacidad para establecer comunicaciones a través de Wi-Fi, Bluetooth y protocolos compatibles con IEEE 802.15.4.
- MSP430, utilizado en dispositivos como los wismote y sky.

Figura 27. Ejemplos de dispositivos Hardware compatibles con ContikiOS

Además, el sistema operativo dispone de un potente simulador llamado Cooja Network Simulator, que junto al emulador MSPSim permite desarrollar y verificar aquellas redes que por su gran tamaño resulten difíciles de testear físicamente. Este simulador hace que esa tarea resulte bastante sencilla ya que proporciona un entorno de simulación que permite emular incluso dispositivos hardware reales en una interfaz gráfica sencilla de utilizar.

2.2.2 OpenWSN

OpenWSN [33] es un proyecto creado en la Universidad de Berkeley en California. Su objetivo es el de construir un estándar abierto basado en open source basado en una pila de protocolos dirigida a las redes de sensores inalámbricos y el Internet de las Cosas (IoT). Está basado en la implementación completa de la pila de protocolos pensada para aplicaciones de WSN, incluyendo las nuevas características del estándar IEEE 802.15.4e como TSCH. Además, es compatible con otros protocolos pensados para el Internet de las Cosas como 6LoWPAN, RPL y CoAP, permitiendo crear redes malladas con dispositivos de ultra-baja-potencia y de alta disponibilidad, que estén completamente conectadas a internet.

Todo este conjunto de herramientas forma parte de un ecosistema creado por la universidad de Berkley dentro del proyecto OpenWSN, contando con dispositivos propios como los OpenMote. El objetivo principal de este proyecto era profundizar en la utilización del estándar IEEE 802.15.4e para crear redes malladas que estuvieran conectadas a internet.

La pila de OpenWSN utiliza dos niveles de abstracción. El Berkeley Socket Abstraction considera que la aplicación se ejecuta en dos hosts de Internet comunicándose a través de un socket, que está identificado únicamente por las direcciones IP de los hosts, y los dos puertos correspondientes de cada aplicación. La pila de OpenWSN respeta esta abstracción, por lo que desarrollar una aplicación en la pila OpenWSN es similar a desarrollar una aplicación en unos hosts de internet comunes.

Por otro lado, tenemos el nivel Hardware Abstraction, que consiste en un grupo de funciones que acceden al hardware en un grupo de archivos llamados Board Support Package (BSP). Esto permite que la mayor parte del código pueda ser compartido por todas las plataformas.

Existen diferentes tipos de sistemas que permiten implementar una pila pensada para redes de tipo WSN, las dos que acabamos de describir son las más conocidas y las que se han utilizado durante el proyecto, pero existen algunas otras soluciones como las que se muestran a continuación.

TinyOS

Es uno de los primeros sistemas operativos creados concretamente para redes WSNs desarrollado por la Universidad de Berkeley. Sus principales características son el uso de un reducido tamaño de memoria, su bajo consumo de energía (bastante optimizado) y la ejecución de operación de concurrencia intensiva (simultaneidad en la ejecución de múltiples tareas interactivas).
El sistema TinyOS está basado en un modelo de programación controlado por eventos en lugar de multiprocesos. Los programas de TinyOS están compuestos por eventos y tareas guiadas, los cuales están escritos en el lenguaje de programación conocido como NesC (extensión del lenguaje de programación C).

El diseño del kernel de TinyOS está estructurado en dos niveles de planificación:

- Eventos: son pequeños procesos pensados para interrumpir las tareas que se están ejecutando. Por ejemplo, cuando el contador del timer (temporizador) se interrumpe, saltará un evento que ejecutará las funciones que se hayan programado para ello.
- Tareas: están realizadas para llevar la carga de procesamiento, por lo tanto, no serán críticas en tiempo. Las tareas, a diferencia de los eventos, se ejecutarán en su totalidad, pero la solicitud de iniciar una tarea y el término de ella son funciones separadas.

El diseño de TinyOS permite que los eventos se ejecuten rápidamente pudiendo interrumpir a las tareas, que llevarán una mayor carga computacional.

MANTIS OS

Es un sistema operativo que surge ante el incremento de la complejidad de las tareas que tienen que realizar los nodos de una red de sensores como son la compresión, agregación y procesamiento de señales.
Los multiprocesos de MantisOS (conocido también como MOS) permiten interpaginar complejas tareas con tareas susceptibles al tiempo, para lograr mitigar los problemas en los saltos de buffers. Para ello, MOS esta implementado para utilizar una pequeña cantidad de memoria RAM, donde con menos de 500 bytes de memoria (incluyendo kernel, controladores de tiempo y pila de comunicación) es capaz de obtener un ahorro de energía bastante considerable, consiguiendo que el microcontrolador solo consuma unos pocos μA cuando pasa a modo sleep tras ejecutar todas las tareas activas.

MantisOS destaca por lo siguiente:

- Flexibilidad en el soporte de múltiples plataformas como PCs, PDAs y diferentes plataformas de microsensores.
- Soporte de control remoto.
- Reprogramación dinámica.
- Acceso remoto.

Nano-RK

Se trata de un sistema operativo totalmente preventivo, basado en la reserva de bajo tiempo real (RTOS) con soporte para redes multisalto y adecuado para implementarse en redes WSN.

Nano-RK es capaz de realizar tareas preventivas con prioridad para asegurar que los plazos de las tareas son conocidos, además de tener un soporte de CPU, red, sensores y actuadores.

Está compuesto por tareas que pueden especificar las demandas de recursos y el SO provee el acceso controlado para los ciclos de CPU y paquetes de las redes, donde todos estos recursos forman la reserva de energía virtual que permite a Nano-RK poder controlar el nivel de energía del sistema y de las tareas.

LiteOs

Es un sistema operativo en tiempo real (RTOS) desarrollado por la Universidad de Illinois para usarlo en aplicaciones de redes de sensores. LiteOS es un sistema operativo basado en Unix que se ajusta a los dispositivos sensores con restricciones de memoria. Este SO permite al usuario operar con WSN como si de un sistema Unix se tratara, lo que lo facilita para personas con conocimientos sobre esta distribución. Proporciona un entorno de programación familiar basado en Unix, threads y C. Sigue un modelo de programación hibrido que permite programar aplicaciones tanto orientada a eventos como a threads. Se trata de un SO de código abierto, desarrollado en C que se puede utilizar en plataformas Atmel AVR basadas en MicaZ e IRIS.

2.3 Selección de hardware

2.3.1 Nodos inalámbricos

Zolertia RE-mote

Dispositivo Zolertia RE-Mote Revisión B. Plataforma de desarrollo hardware inalámbrico diseñada para productos IoT, proporcionando una solución de hardware resistente a la industria. Producto de ultra bajo consumo que permite al dispositivo ser autónomo con una pequeña batería. Sus características principales son:

- Plataforma:
  - Dos transceptores para comunicación inalámbrica: 863 MHz y 2.4 GHz. Siendo posible actuar de forma simultánea con ambas frecuencias.
  - Real-Time Clock.
  - Bloque Power Management que reduce hasta 150nA el consumo al despertar el real-time clock.
  - Cargador de batería on-board.
  - Alimentación desde 3.5VDC hasta 16VDC soportando alimentación desde panel solar.
  - Slot para microSD.
  - Medidor externo de batería vía I2C.
  - Programable vía USB.
  - RGB LED on-board, botón de reset, botón configurable y un botón opcional para el master reset.
  - USB 2.0 (12Mbps).
- Microcontrolador:
  - CC2538 ARM® cortex®-M3.
  - Velocidad de reloj mayor a 32MHz.
  - 512KB flash programable.
  - 32KB de RAM (16KB de retención en todos los power modes).
  - JTAG Debugging.
  - Low-Power:
    - Active mode: 20mA.
    - Power Mode 1 (4μs wake-up): 6mA.
    - Power Mode 2 (sleep timer running and 16KB retention): 1.3μA.
    - Power Mode 3 (External Interrupt and 16KB retention): 0.4μA.
- Radio:
  - ISM 2.4GHz IEEE 802.15.4 Compliant transceiver:
    - Sensibilidad del receptor: -97dBm.
    - Robustez frente a interferencias con ACR 44dB.
    - Programable potencia de salida >7dBm
    - 250Kbps data rate con modulación DSSS.
    - Recepción (CPU idle): 20 mA de pico.
    - Transmisión (CPU idle, @0dBm): 24mA de pico.
  - ISM 863-950MHz IEEE 802.15.4 Compliant transceiver:
    - Sensibilidad del receptor: -123dBm @1.2kbps, -109dBm @50kbps.
    - Bloqueo 86dBm en 10MHz.
    - Sensibilidad canales adyacentes: >60dB en un offset de 12.5Khz.
    - Programable potencia de salida >16dBm.
    - Tipos de modulación: 2-FSK, 2-GFSK, 4FSK, 4-GFSK, MSK, OOK.
    - Data rate > 1.25Mbps.
    - Bajo consumo: 0.12μA, 0.5μA al encender radio (eWOR).
    - Recepción: 19mA, 0.5mA en RX Sniff Mode.
    - Transmission (@10-14dBm): 35-46mA.

Figura 28. Zolertia RE-Mote

OpenMote CC2538

Se trata de una plataforma de prototipado modular para comunicaciones inalámbricas IoT como las WSN. La idea principal del ecosistema OpenMote es separar el módulo de comunicaciones y computación del interfaz de la placa pudiendo así utilizar diferentes módulos con distintos propósitos como el OpenBattery, el OpenBase o el OpenUSB, siendo el módulo OpenMote-CC2538 el corazón de este ecosistema.

- Microcontrolador SoC CC2538 de Texas Instrument de 32-bits Cortex-M3.
- Su radio opera a 2.4 GHz y es compatible con IEEE 802.15.4-2006.
- Velocidad de reloj de hasta 32 MHz.
- Utiliza una memoria RAM de 32KB y una memoria flash de 512KB.
- Incluye diferentes periféricos como: GPIOs, ADC, I2C, SPI, UART.
- El subsistema de potencia se lleva a cabo por un conversor DC/DC (TPS62730)
- Dispone de 4 LEDs, 2 botones programables, una antena de montaje superficial y también un conector SMA para colocar una antena externa.
- Compatible con OpenWSN y Contiki.

Figura 29. OpenMote-CC2538.

Zolertia Z1

La plataforma Z1 de Zolertia es una de las versiones anteriores a los actuales RE-Mote. Se trata de una plataforma de desarrollo de propósito general diseñada para que los investigadores trabajen en las WSN. Es una plataforma compatible con la familia Tmote añadiendo algunas mejoras que ofrecen un mejor rendimiento.

- Microcontrolador MSP430F217 de bajo consumo.
- CPU con una velocidad de 16MHz
- Memoria RAM de 8KB y memoria Flash de 92KB
- Transceptor de baja potencia CC2420 en la banda de 2.4GHz con una tasa de 250Kbps.
- Soporte con el estándar IEEE 802.15.4 (No soporta las últimas versiones con TSCH).
- Compatibilidad con protocolos IP, así como 6LoWPAN.
- Sensores digitales integrados (temperatura y acelerómetro 3-ejes)

Figura 30. Zolertia Z1.

2.3.2 Gateway

Raspberry PI 3

Computador de placa de bajo coste. Cada vez más conocido en entornos cuotidianos, son dispositivos con gran capacidad de computación y de comunicaciones. Tiene una amplia comunidad que respalda todo tipo de proyectos y solución de problemas. Sus posibilidades de conectividad le otorgan un puesto privilegiado en proyectos IoT. Sus características principales son:

- Microcontrolador:
  - Broadcom BCM2837 a 1.2GHz
  - 64bits-Quad-core ARM® cortex®-A53.
- Plataforma:
  - 1GB de memoria LPDDR2.
  - Puerto 10/100 Ethernet.
  - 802.11 b/g/n Wireless LAN.
  - Bluetooth 4.1 (Clásico y Low Power).
  - Conector microUSB para alimentación hasta 2.5A.
  - Conector HDMI video y audio.
  - Conector RCA video y audio.
  - Conector CSI cámara.
  - 4 x USB.
  - 40 GPIO pins.
  - Slot microSD.
- Comunicaciones
  - Ethernet.
  - Bluetooth.

Figura 31. Raspberry pi 3

BeagleBone Black

Se trata de un dispositivo de bajo coste con una extensa comunidad que respalda la plataforma y a los desarrolladores. Está basado en un procesador SITARA de Texas Instruments. Sus principales características son:

- Procesador:
  - Sitara ARM® cortex® - A8 32-Bits. 1GHz, 2000 MIPS.
  - 64kb de memoria RAM interna.
  - Soporte de protocolos de comunicación de ámbito industrial, así como EtherCAT®, PROFIBUS, PROFINET, EtherNet/IP™ entre otros.
  - Dos PRUs (Programmable Real-Time Units).
  - Reloj de tiempo real.
- Plataforma:
  - 512MB DDR3 RAM.
  - 4GB 8bit eMMC almacenamiento flash on-board.
  - 3D graphics accelerator.
  - 2 x PRU 32-bit microcontrollers.
- Software compatible:
  - Debian
  - Android
  - Ubuntu
  - Cloud9
- Conectividad:
  - USB cliente para comunicaciones y alimentación.
  - USB host.
  - Ethernet.
  - HDMI.
  - 2x46 pines.
- Conectores de expansión:
  - Power: 5V, 3.3V, VDD_ADC(1.8V)
  - 3.3V I/O en todas las señales McASP0, SPI1, I2C, GPIO(69max), LCD, GPMC, MMC1, MMC2, 7AIN (1.8V Max), 4 Timers, 4 Puertos serie, CAN0, EHRPWM (0,2), XDMA interrupciones, botón de alimentación.

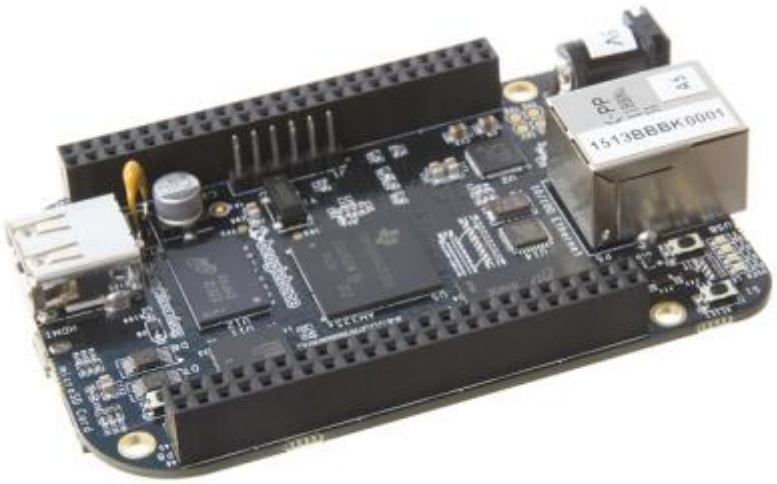


Figura 32. BeaggleBone Black

Capítulo 3. Metodología

3.1 Realización del proyecto

La realización de este proyecto se ha llevado a cabo mediante la realización de prácticas en empresas en el Instituto Tecnológico de Informática (ITI) durante un periodo de 9 meses. En este proyecto se recogen la mayoría de las actividades realizadas durante este periodo de tiempo, desde la elaboración de recopilación bibliográfica hasta las campañas de medidas que han servido para aportar algunos de los resultados recogidos en este documento.

Al inicio de las prácticas, aunque contaba con algunos conocimientos básicos sobre redes de sensores, hizo falta una profundización más exhaustiva sobre algunas de las tecnologías de comunicaciones robustas para redes de sensores inalámbricas (WSN). Para ello se realizó un estudio de los diferentes protocolos con los que el grupo de investigación se encontraba trabajando en ese momento, entre los que se encuentran el método de acceso de recursos MAC TSCH que forma parte del estándar IEEE 802.15.4 y el protocolo de encaminamiento dinámico orientado a redes LLN que es RPL. Esto sirvió para elaborar un documento que recogiese algunas de las características más destacadas de estos protocolos con el objetivo de aclarar y poner ideas en común dentro del grupo de investigación, lo que serviría para escoger las líneas de investigación que se afrontarían de cara a estos protocolos.

Una vez familiarizado con la pila de protocolos con la que construiríamos las redes de sensores a nivel teórico, se empezaron a realizar pruebas reales y a familiarizarse con el entorno de desarrollo, que sería el sistema operativo ContikiOS y los dispositivos hardware que el grupo de investigación acababa de adquirir recientemente, los RE-Mote de Zolertia. Para esto se realizaron diferentes modificaciones de algunos de los ejemplos incluidos en Contiki para observar las capacidades que este ofrecía, realizando tanto simulaciones mediante el software Cooja incluido en Contiki como en implementaciones reales en los dispositivos embebidos de Zolertia.

Durante este periodo, se realizaron diferentes trabajos de documentación para recoger las principales ideas sobre las herramientas, protocolos o sistemas con los que íbamos a trabajar. Entre estos trabajos se encuentran un resumen sobre los métodos y algoritmos de planificación de los recursos de TSCH, la comparación de los principales protocolos industriales (WirelessHART, ISA100.11a, etc) con el estándar en el que se basan (IEEE 802.15.4) con el fin de poder comparar sus diferentes características. También se recopilaron diferentes mecanismos y herramientas que ayudasen en la fase de despliegue de las WSN, utilizados tanto por el sector industrial como otros implementados por la comunidad científica.

También se realizaron pruebas de campo y simulaciones de diferentes soluciones desarrolladas junto al grupo de investigación que servirían para la presentación de diferentes artículos de investigación que se presentarán en revistas y congresos. En concreto se ha realizado un artículo de revista que ya ha sido aceptado y otro para congreso que se presentará en los próximos meses

3.2 Distribución, seguimiento y diagrama temporal

En este apartado se muestra un listado de las tareas realizadas a lo largo del desarrollo de las prácticas y del proyecto. Además, se indican las fechas aproximadas en las que se desempeñaron dichas tareas con el fin de realizar un diagrama temporal. Ya que se trataba de prácticas en empresas la dedicación estaba bastante marcada por el horario laboral que se seguía, siendo una dedicación de 4 horas diarias durante los 6 primeros meses y de 8 horas diarias durante los 3 meses restantes.

Tarea 1. Búsqueda y recopilación de información.

1.1. Búsqueda y lectura de información acerca de los diferentes protocolos utilizados en las WSN, entre los que se encuentran algunos como TSCH y RPL. (7/11/16 – 9/11/16)

1.2. Familiarización con el entorno de desarrollo de ContikiOS. Contempla tanto la búsqueda de información de los sistemas que intervienen como las diferentes modificaciones de ejemplos para testear su funcionamiento. (8/11/16 – 11/11/16)

1.3. Búsqueda de diferentes métodos de planificación de los recursos MAC de TSCH. (9/1/17 – 20/1/17)

1.4. Recopilación de información sobre los protocolos industriales actuales y de su funcionamiento en las WSN. Entre estos se encuentran WirelessHART, ISA100.11a, ZigBee Pro y WIA-PA. (9/3/17 – 11/4/17)

1.5. Búsqueda bibliográfica sobre sistemas o herramientas que den soporte a la etapa de despliegue de una WSN. (18/4/17 – 21/6/17)

Tarea 2. Elaboración de documentos bibliográficos y artículos de investigación.

2.1. Comparación de diferentes algoritmos y mecanismos de planificación de recursos TSCH, mostrando algunos ejemplos de mecanismos centralizados, distribuidos y el mecanismo denominado como autónomo Orchestra. (16/1/17 – 25/1/17)

2.2. Comparación de diferentes aspectos relacionados con los protocolos industriales WirelessHART, ISA100.11a, ZigBee Pro y WIA-PA. (22/3/17 – 21/4/17)

2.3. Elaboración de un documento que recopile diferentes mecanismos que ayuden en la etapa de despliegue de las WSN, diferenciándolos en pre-despliegue, despliegue y post-despliegue. (22/6/17 – 2/8/17)

2.4. Elaboración de un artículo de revista mostrando los resultados de las pruebas de energía. (Antes de Noviembre – Mayo que se envió a la editora)

2.5. Elaboración de un artículo para congreso mostrando los resultados de las simulaciones de calidad del enlace. (19/7/17 – X/9/17)

Tarea 3. Desarrollo de los programas necesarios para las pruebas de energía.

3.1. Utilización de la herramienta Powertrace para obtener las medidas de energía. (14/11/16 – 18/11/16)

3.2. Modificación de la función objetivo MRHOF para que utilice parámetros de energía en lugar de la métrica ETX. (16/11/16 – 30/11/16)

Tarea 4. Campaña de medidas de las pruebas de energía.

4.1. Búsqueda de la localización donde se realizarían las pruebas, probando diferentes valores de potencia de transmisión. (23/11/16)

4.2. Realización de las pruebas de campo utilizando la función objetivo MRHOF. (24/11/17 – 19/1/17)

4.3. Realización de las pruebas de campo utilizando la función objetivo que se ha desarrollado utilizando la información de la energía consumida por las baterías. (24/11/17 – 19/1/17)

4.4. Comparación de resultados y elaboración de las conclusiones. (16/1/17 – 20/1/17)

Tarea 5. Desarrollo de los programas necesarios para las pruebas sobre calidad de los enlaces.

5.1. Creación de una estructura de datos donde poder almacenar las métricas utilizadas para la función de calidad y obtención de las mismas. (24/2/17 – 26/3/17)

5.2. Definición de las funciones de mapeo para cada una de las métricas y obtención de la función de calidad. (8/3/17 – 12/3/17)

5.3. Definir los límites de calidad que se utilizarán en la función objetivo para determinar cuál será el mejor camino. (25/3/17 – 7/4/17)

Tarea 6. Realización de las simulaciones para las pruebas sobre calidad de los enlaces.

6.1. Selección del modelo de simulación utilizado para estas pruebas. (12/6/17)

6.2. Desarrollo de un pequeño programa que controle los parámetros de la simulación durante el tiempo de ejecución del mismo. (13/6/17 – 15/6/17)

6.3. Realización de las simulaciones utilizando la función objetivo MRHOF. (15/6/17 – 31/8/17)

6.4. Realización de las simulaciones utilizando la función objetivo que se ha desarrollado utilizando la función de calidad y dando unos pesos a cada uno de los parámetros de manera individual. (15/6/17 – 31/8/17)

6.5. Realización de las simulaciones utilizando la función objetivo que se ha desarrollado utilizando la función de calidad y dando unos pesos similares a cada uno de los parámetros. (15/6/17 – 31/8/17)

6.6. Comparación de resultado y elaboración de las conclusiones. (28/8/17 – 31/8/17)

Tarea 7. Configuración de diferentes tipos de planificaciones de recursos TSCH.

7.1. Modificación del método de planificación Minimal 6TiSCH. (6/2/17 – 7/2/17)

7.2. Configuración del mecanismo de planificación autónomo Orchestra. (8/2/17 – 10/2/17)

Tarea 8. Simulaciones de diferentes tipos de planificaciones de recursos TSCH.

8.1. Realización de las simulaciones utilizando la planificación modificada de Minimal 6TiSCH. (13/2/17 – 28/2/17)

8.2. Realización de las simulaciones utilizando la planificación proporcionada por el mecanismo de Orchestra. (13/2/17 – 28/2/17)

Tarea 9. Elaboración de la memoria del trabajo.

9.1. Redactar el Capítulo 1: Introducción, objetivos y estructura de la memoria. (20/6/17)

9.2. Redactar el Capítulo 2: Estado del arte y selección de tecnologías. (20/6/17 – 31/8/17)

9.3. Redactar el Capítulo 3: Metodología. (15/7/17)

9.4. Redactar el Capítulo 4: Desarrollo y resultados. (11/7/17 – 31/8/17)

9.5. Redactar el Capítulo 5: Conclusiones y propuestas de trabajo futuro. (31/8/17)

9.6. Realización y preparación de la presentación. (1/9/17 – 19/9/17)

A continuación, se puede ver el diagrama de Gantt con la distribución de las tareas comentadas.

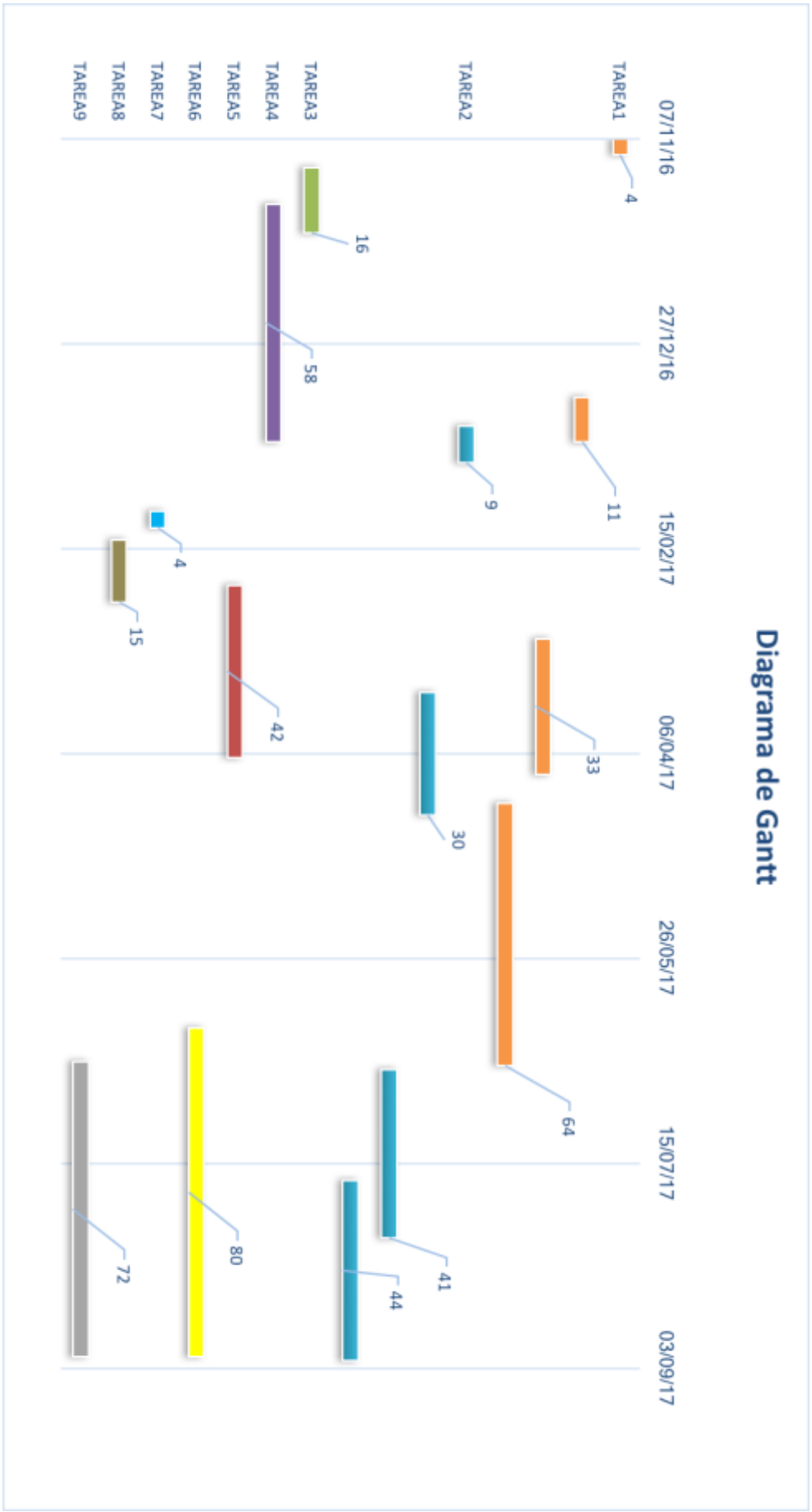


Figura 33. Diagrama de Gantt de las tareas realizadas.

3.3 Presupuesto

Para el desarrollo de este proyecto y la realización de las prácticas el grupo de investigación disponía de diferentes kits de desarrollo y diferentes dispositivos que se han empleado para avanzar en las diferentes líneas de investigación, algunos de los cuales se incluyen en el siguiente presupuesto.

Producto	**Cantidad**	**Precio unitario**	**Precio total**
RE-Mote Research Pack de Zolertia	2	1489,95€	2979,9€
Raspberry Pi 3	1	49,95€	49,95€
BeagleBone Black – Rev C	1	51,00€	51,00€
Precio total productos:			**3080,85€**

Tabla 3. Tabla de presupuesto del proyecto.

El RE-Mote Research Pack incluye los siguientes productos:

- 15 x dispositivos RE-Mote para IoT.
- 15 x baterías LiPo de 600 mAH @ 3.7V.
- 15 x antenas multi banda.
- 15 x encapsulados de plástico para los RE-Mote de Zolertia.
- Conectores Phidgets (x2) y Ziglet (x1) en cada RE-Mote.
- 8 x cables micro USB.

Con este número de dispositivos podemos desplegar redes de hasta un total de 30 nodos utilizando cualquiera de los dispositivos (RaspberryPi o Beaggle Bone) como Gateway. En este presupuesto no se han incluido los ordenadores necesarios para configurar los diferentes dispositivos.

Capítulo 4. Desarrollo y resultados

En este capítulo se explicarán todos los pasos que se han llevado a cabo para el diseño de las diferentes pruebas y simulaciones. El capítulo estará dividido en tres partes, que se corresponderán con tres diferentes implementaciones que se han realizado durante el desarrollo de este proyecto. Para cada una de estas fases se detallarán los procesos llevados a cabo para desarrollar la implementación, así como una explicación de las pruebas o simulaciones, finalizando con una breve explicación de los resultados obtenidos en cada una de ellas.

4.1 Puesta en marcha y primeros pasos en ContikiOS

Al inicio de este proyecto se realizaron unas pequeñas tareas de documentación para aprender las bases tanto de los protocolos como de los sistemas con los que trabajaríamos en el resto de implementaciones llevadas a cabo por el grupo de investigación del ITI. En el Capítulo 2 se han mostrado algunas de las posibles soluciones en cuanto a dispositivos hardware y entornos de desarrollo que se encuentran disponibles para trabajar en el área de las WSN. Durante este proyecto se utilizarán los dispositivos RE-Mote de Zolertia ya que se trata de una plataforma bastante completa, que permite utilizar diferentes frecuencias radio, ya sea de manera individual como de forma conjunta, permitiendo así tener una mayor flexibilidad a la hora de trabajar con ellos. Además, las distintas implementaciones que se llevarán a cabo estarán basadas en el método de acceso al medio TSCH, por lo que esta característica deberá ser soportada por la plataforma hardware, siendo el caso de los dispositivos RE-Mote.

Por otro lado, el entorno de desarrollo que utilizaremos será el sistema operativo ContikiOS, que además de ser compatible con los RE-Mote, permiten construir una estructura de protocolos de manera eficiente, resultando de utilidad algunas de sus características como la utilización de diferentes hilos de ejecución y el sistema de eventos, que nos facilitará la tarea de implementar nuestras funciones en la pila de Contiki.

En este apartado se detallarán algunos de los conocimientos básicos necesarios para utilizar el sistema operativo de Contiki y poder implementar dichas aplicaciones en un dispositivo hardware real como los RE-Mote, además de describir la forma en la que se realizan las simulaciones en Cooja. El sistema de Contiki se puede obtener junto a un grupo de herramientas ya configuradas para poder trabajar con él en una distribución Linux de Ubuntu. Este grupo de herramientas viene recogido en el paquete de instalación Instant Contiki. De esta forma lo único que debemos hacer es instalar la distribución de Ubuntu que nos proporciona Contiki, ya sea directamente sobre el sistema del ordenador o bien mediante una máquina virtual. En nuestro caso utilizaremos una máquina virtual configurada en VMWare. La versión de Contiki que instalaremos será la 3.0 que es la más reciente hasta ahora. Una vez instalado, dentro de los directorios en Ubuntu encontraremos un conjunto de carpetas de Contiki donde reside el sistema (Figura 34). Aunque se supone que hemos instalado la última versión del entorno, la 3.0, es conveniente reemplazar este directorio de carpetas por las que se incluyen en el repositorio de

Contiki en Github, ya que la comunidad de desarrolladores está continuamente mejorando el sistema con nuevas características.

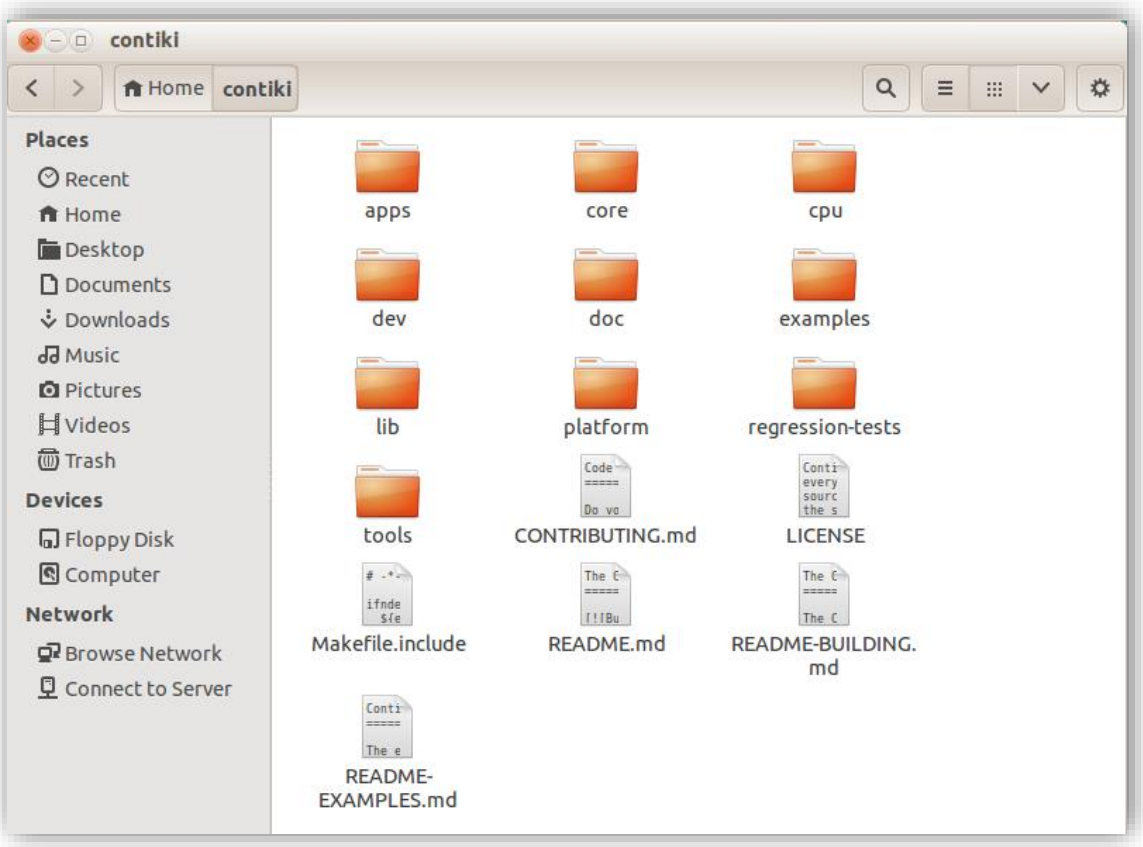


Figura 34. Directorio de ContikiOS

En la figura anterior podemos ver cómo están estructuradas las carpetas en Contiki:

- **Apps**: en este directorio se recogen las diferentes aplicaciones que se pueden utilizar como programas auxiliares dentro del programa principal que contendrá la pila de protocolos. Entre algunas de las aplicaciones más destacadas que se encuentran en este directorio podemos encontrar JSON, MQTT o el mecanismo de planificación de recursos Orchestra.
- **Core**: esta carpeta recoge todo el sistema operativo, incluye toda la pila de protocolos, sistemas para implementar los threads, temporizadores y el resto de características necesarias para implementar el sistema en un dispositivo externo.
- **CPU**: en este directorio se encuentra todo el conjunto de archivos necesarios para implementar el sistema en cada uno de los microcontroladores soportados. Nos puede servir para modificar características sobre los temporizadores, gestionar la memoria flash, modos de funcionamiento de bajo consumo, etc. En nuestro caso, ya que utilizaremos los RE-Mote deberemos acceder a la carpeta cc2538 que es el que utilizan los nodos de Zolertia. De esta forma podremos modificar algunos parámetros como es la potencia de transmisión radio.
- **Doc**: se trata de un conjunto de archivos de documentación sobre el sistema.
- **Examples**: en esta carpeta podemos encontrar multitud de programas desarrollados por la comunidad de Contiki y que permiten verificar algunos ejemplos como comunicaciones IP, etc.
- **Platform**: aquí podemos encontrar todo el conjunto de archivos que dependerá de la plataforma que utilicemos, en nuestro caso los RE-Mote utilizan la plataforma zoul que se encuentra en este directorio y que permite configurar algunas de sus características.

- **Tools**: la última de las carpetas incluye diferentes herramientas que no forman parte del sistema operativo de Contiki pero que pueden servir para realizar tareas de depuración o simulación. Es aquí donde podremos encontrar el Cooja Network Simulator para realizar nuestras simulaciones.

Para nuestras diferentes aplicaciones realizaremos modificaciones sobre todo de diferentes ejemplos ya planteados y sobre algunas características de la pila de protocolos. Esta pila de protocolos se encuentra en la carpeta net dentro del directorio core. Para poder realizar modificaciones en nuestras aplicaciones deberemos conocer la estructura de programación que se utiliza en Contiki, cuyos programas están desarrollados en lenguaje C.

La estructura de los programas en Contiki sigue la siguiente distribución.

```
/* Declaración de los archivos de cabecera */
#include “contiki.h”
#include <stdio.h>

PROCESS(example_app, “Aplicación de ejemplo”);
AUTOSTART_PROCESSES(&example_app);

PROCESS_THREAD(example_app,ev,data)
{
      /* Se declaran las variables necesarias */
      static int i = 15;

      /* Comienzo del proceso */
      PROCESS_BEGIN();

      /* Conjunto de declaraciones en C */
      printf(“El numero almacenado en i es %d\n”,i);

      /* Termina el proceso */
      PROCESS_END();
}
```

Este es un ejemplo muy sencillo de lo que sería un programa desarrollado en C, pero permite mostrar la estructura que siguen el resto de ejemplos. Cada uno delos procesos que estén presentes en el programa deben iniciarse con el macro PROCESS, en nuestro caso solo tenemos uno por lo que le pasamos los argumentos de la variable donde se almacenará la estructura y un nombre descriptivo del proceso. Una vez hecho esto, se utiliza el macro AUTOSTART_PROCESS para que el proceso que le indiquemos como argumento comience automáticamente.

A continuación, se definen los diferentes threads por los que estará formada nuestra aplicación, definiendo una variable de evento que podrá ser llamada cada vez que se produzca un evento determinado en el sistema. Después, utilizando los macros PROCESS_BEGIN y PROCESS_END podremos delimitar cuando empieza y cuando acaba el proceso, que será donde estarán las sentencias de código en C que determinarán el funcionamiento del programa. En este caso únicamente se mostrará un mensaje con la variable que se ha definido fuera del proceso.

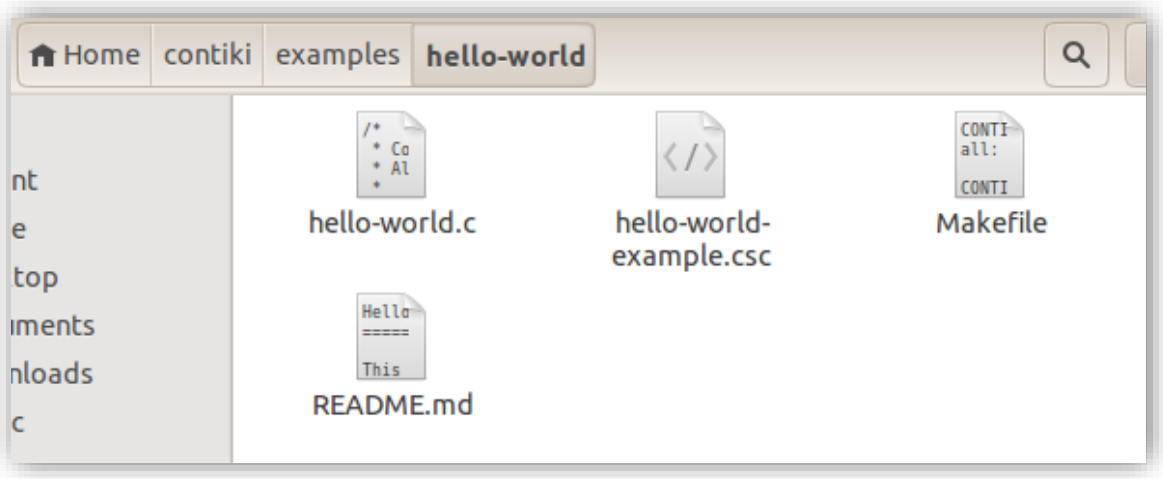


Figura 35. Ejemplo de aplicación hello-world en Contiki

Además del archivo que contiene el programa es necesario crear otro archivo llamado Makefile dentro del directorio del programa. Este archivo nos permitirá incluir todo el sistema de archivos del sistema operativo para que funcione de manera conjunta a nuestro programa, así como servir para definir algunas variables de configuración. Además, este archivo nos servirá para incluir algunas características a nuestra pila de protocolos como el modo TSCH o para incluir aplicaciones externas a nuestros programas como por ejemplo Orchestra.

Adicionalmente, se deberá incluir un archivo que contenga la configuración del proyecto llamado project-conf.h, desde el que podremos modificar diferentes configuraciones de la pila de protocolos o de otras aplicaciones mediante sus diferentes parámetros de configuración.

4.1.1 Método para cargar los programas en los RE-Mote

Una vez tengamos nuestra aplicación desarrollada lo único que quedará será cargarla en la plataforma hardware en la que queramos testearlo. Los dispositivos RE-Mote disponen de dos puertos micro USB directamente ensamblados en la placa, uno para las tareas de programar y depurar y el otro para poder ser utilizado como puerto USB en alguna aplicación. La carga de los programas se realiza a través de este convertidor serie a USB lo que nos permitirá transferir nuestro código binario una vez esté compilado, donde este lo ejecutará una vez el dispositivo se resetee.

Figura 36. Esquema de los diferentes puertos y botones del RE-Mote [34].

Para cargar nuestro programa lo único que tendremos que hacer es conectar el RE-Mote al ordenador mediante el cable microUSB y abrir un terminal en el sistema Ubuntu. Desde el terminal accederemos al directorio donde tengamos almacenado nuestro programa y ejecutaremos el código que se muestra en la Figura 37. De esta manera conseguimos compilar todos los archivos necesarios del sistema ContikiOS, además de nuestro programa. Para cargarlo basta con sustituir el nombre del fichero por broadcast-example.upload para el caso que se muestra en la figura. El sistema borrará la memoria actual de los RE-Mote y cargará el nuevo archivo binario que se acaba de compilar. Mediante la declaración del TARGET definimos la plataforma en la que se va a cargar el programa, en nuestro caso esa plataforma será zoul que sirve para los RE-Mote y para otros dispositivos hardware como los Firefly.

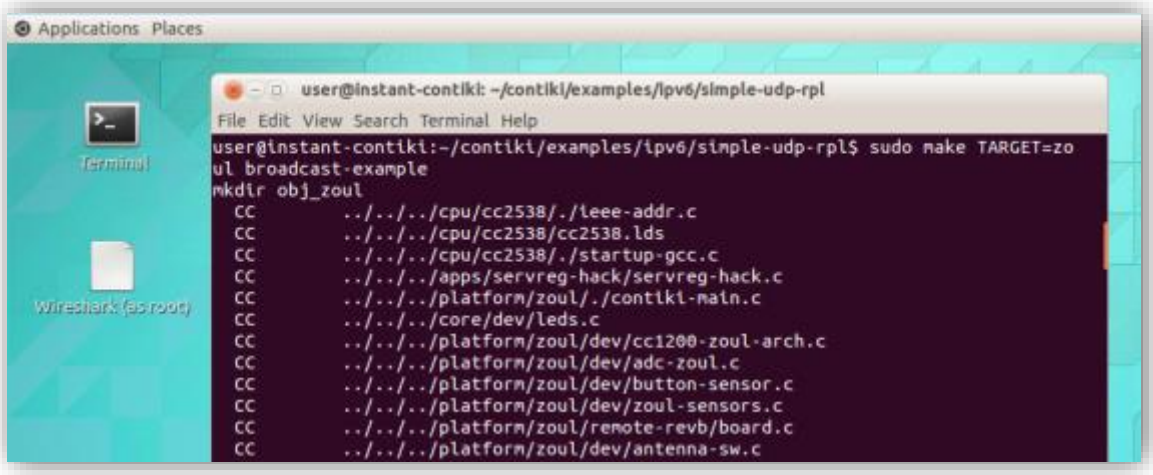


Figura 37. Compilando el código para los RE-Mote

4.1.2 Simulación en Cooja de una WSN emitiendo mensajes broadcast

A continuación, vamos a realizar una pequeña simulación de uno de los ejemplos que se incluyen en Contiki. Esto nos permitirá explicar el funcionamiento del Cooja Network Simulator, mostrando sus diferentes opciones de configuración, así como las características que nos puede ofrecer a la hora de simular una WSN. En concreto crearemos una red muy sencilla en la que los nodos estén transmitiendo mensajes broadcast a todos sus vecinos.

Para poder ejecutar Cooja es necesario tener instalada al menos la versión 1.6 de Java, así como la herramienta *ant* que permite montar la plataforma. Para compilar y ejecutar Cooja solo deberemos acceder al directorio donde se encuentra la aplicación y ejecutar los siguientes comandos:

```
>> cd contiki/tolos/cooja
>> ant run
```

El entorno de Cooja se compilará y comenzará a funcionar la aplicación. Una vez tengamos la aplicación abierta nos aparecerá un menú superior desde el que podremos crear nuevas simulaciones, controlar los estados de la simulación, utilizar diferentes tipos de herramientas o configurar cada uno de los nodos que formarán parte de la simulación.

Para empezar, deberemos crear una nueva simulación, o en su defecto abrir una ya existente. Los archivos de simulación tienen una extensión '.csc' y muchos de los ejemplos incluidos en Contiki disponen de archivos de simulación ya configurados.

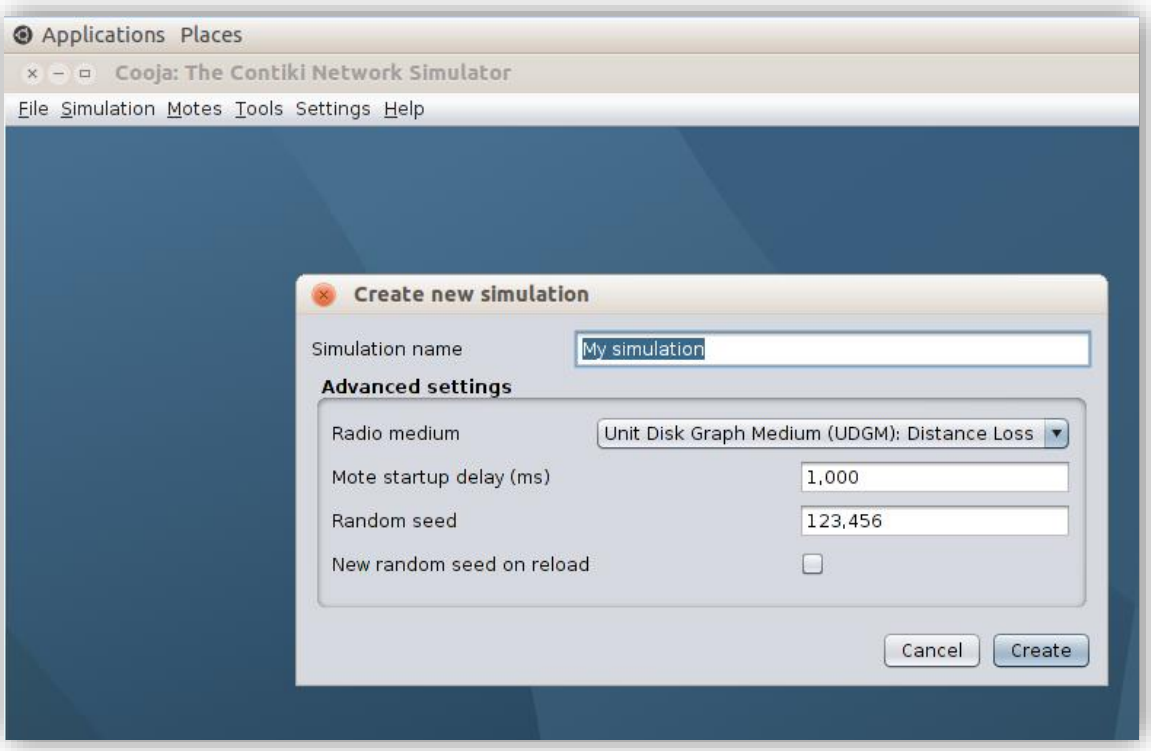


Figura 38. Creando una nueva simulación en Cooja.

En la Figura 38 podemos ver la ventana emergente que aparece al crear una nueva simulación. En ella podremos configurar algunos aspectos del modelo de propagación radioeléctrico, el retardo que les llevará a los nodos para iniciarse o la configuración de una semilla aleatoria que permitirá realizar diferentes simulaciones dentro de una distribución de probabilidad. Si quisiéramos obtener siempre el mismo resultado, con mismos paquetes perdidos, etc., utilizaríamos una misma semilla.

En cuanto a los modelos de propagación de Cooja, estos no están muy bien documentados [35] por lo que es necesario revisar sus características directamente en el código fuente, donde se incluyen algunos comentarios que pueden aclarar su funcionamiento. Los modelos que presenta Cooja son los siguientes:

- **No Radio Traffic**: Este modelo no tiene mucho que destacar ya que no resulta de interés para simular WSN debido a que no se utiliza un medio de comunicación inalámbrico.
- **Unit Disk Graph Medium (UDGM) – Constant Loss**: Este es un modelo muy sencillo en el que el rango de transmisión se modela como una circunferencia ideal que representa el rango de cobertura del nodo, cualquier otro nodo que se encuentre fuera de esta circunferencia no se podrá comunicar directamente con él, mientras que los nodos dentro del rango de transmisión podrán recibir todos los paquetes que esté enviando. El rango máximo de transmisión se multiplicará por la fracción de potencia de salida respecto a la máxima potencia de salida que pueda tener el dispositivo simulado y el resultado de la potencia de transmisión se compara con la distancia en el escenario de simulación. Esto significa que, si el rango máximo de transmisión de un dispositivo es de 100 m y la potencia actual de salida es la mitad del máximo posible, la circunferencia dentro de la cual los paquetes se recibirán correctamente tendrá un radio de 50 m.
- **Unit Disk Graph Medium (UDGM) – Distance Loss**: Esta es una nueva implementación del modelo UDGM anteriormente descrito, pero ampliando sus caracteristcas de dos formas. Primero, se consideran las interferencias, pero de una

manera sencilla, si el paquete que se pretende transmitir se interfiere, el paquete se perderá. Esto significa que todas las comunicaciones que se lleven a cabo en el mismo instante de tiempo no se realizarán con éxito, cosa que difiera con la realidad. La segunda característica es que es posible definir un ratio o porcentaje de éxito en transmisión y recepción. Esto permitiría controlar la probabilidad de perder paquetes en un enlace.

- **Directed Graph Radio Medium (DGRM)**: Este modelo se puede utilizar de manera independiente o como base para otras simulaciones. El propósito de este tipo de medio radio es poder especificar las características de los enlaces de manera asimétrica, pudiendo controlar el porcentaje de perdida de paquetes, el valor de RSSI, la calidad del enlace LQI o el delay. Sin embargo, no contempla las perdidas por distancia, por lo que en ocasiones puede resultar interesante combinarlo con otro medio de propagación.
- **Multi-path Ray-tracer Medium (MRM)**. Este modelo es una nueva alternativa implementada en Cooja como modelo de propagación de la señal. Este modelo no está basado en medidas empíricas si no en una aproximación analítica. Este modelo utiliza la técnica de trazado de rayos modelada en dos dimensiones. De esta forma la potencia recibida se calcularía mediante la fórmula de Friss, pero teniendo en cuenta diferentes obstáculos que pueden definirse en el entorno de simulación como elementos atenuadores. Se calculan valores de refracción, reflexión y difracción por lo que proporciona una solución mucho más realista, pero que necesita de una mayor capacidad de computación.

Para realizar este ejemplo se ha seleccionado un modelo sencillo pero que tiene en cuenta las posibles interferencias que se puedan producir, el modelo UDGM: Distance Loss, tal y como se muestra en la Figura 39.

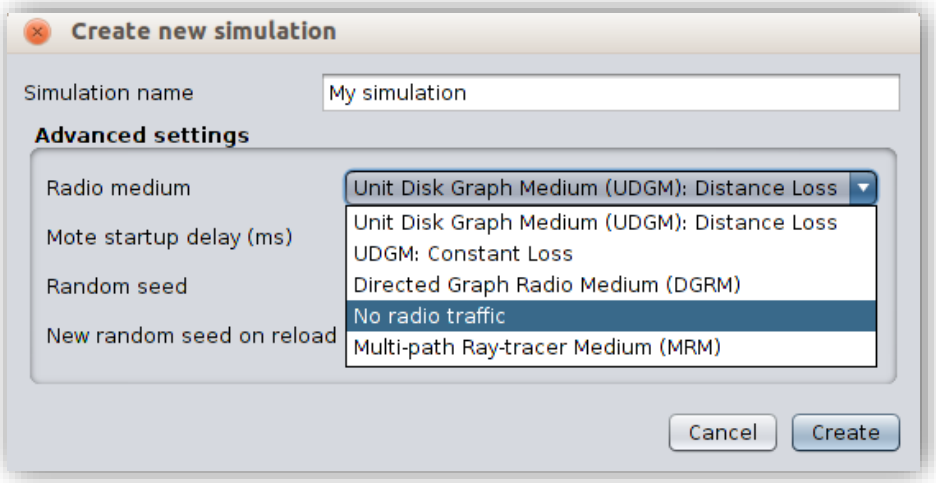


Figura 39. Tipos de medios radio en Cooja

Una vez creada la nueva simulación se nos abrirá el entorno de simulación con una serie de herramientas que nos permitirán controlar los valores de simulación, construir la topología de la red o mostrar los mensajes recibidos por los diferentes nodos. Lo primero que hay que hacer es definir los tipos de nodos que se van a utilizar en la simulación mediante la pestaña Motes. En esta pestaña se nos permite elegir el tipo de nodo que queremos simular, pudiendo elegir entre un mote de tipo Cooja, que estará controlado completamente por el simulador y que utilizará una plataforma nativa. Por otro lado, podemos escoger otras plataformas como los Wismote, Sky mote o Z1 mote entre algunas de las posibles opciones. Estos últimos ejemplos se añadirán a la simulación como dispositivos emulados por lo que tendrán un comportamiento más aproximado al de la realidad, aunque la velocidad de la simulación se verá mermada en comparación con los Cooja mote simulados. Una vez escogido el tipo de nodos podremos

buscar el programa que queramos cargar en los diferentes dispositivos para compilarlo y configurarlo en los nodos, tal y como muestra la Figura 40.

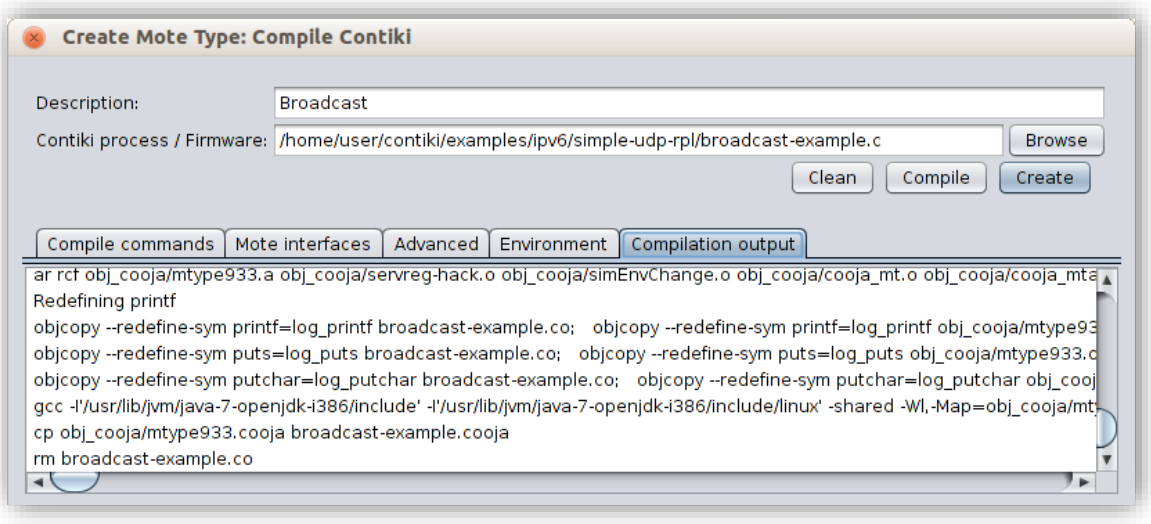


Figura 40. Compilación de la aplicación en Cooja.

Si el programa se compila correctamente podemos proceder a crear los nodos, momento en el que nos aparecerá una ventana emergente como la de la Figura 41 en el que podremos indicar el número de nodos con la misma configuración que queramos colocar, así como la posición en la que se deben distribuir los nodos que elijamos, pudiendo distribuirse de manera aleatoria, lineal, en una elipse o también está la posibilidad de colocarlos manualmente si queremos realizar una topología más precisa.

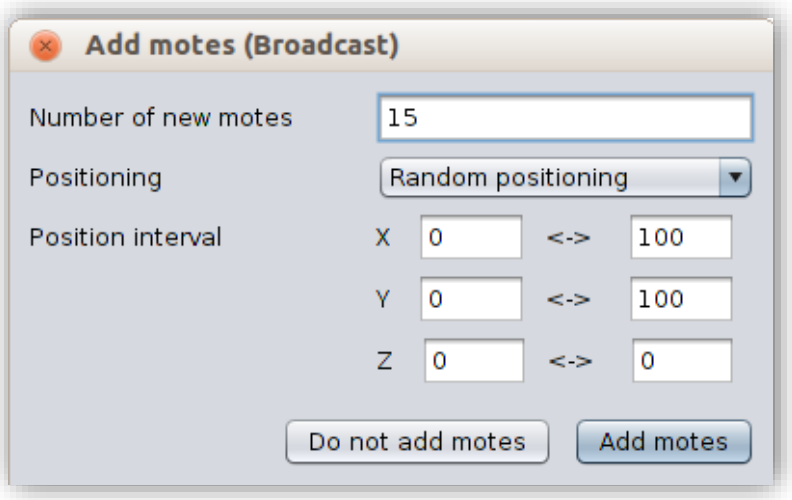


Figura 41. Añadir los nodos a la simulación.

Una vez añadidos los nodos podemos volver a crear otro tipo de nodo si la aplicación lo requiere y una vez tengamos colocados todos ellos, se podrá comenzar con la simulación. Entre las herramientas que se pueden utilizar en el entorno de simulación dentro de la pestaña Tools, podemos describir un conjunto que pueden resultar más interesantes. Estas herramientas aparecen como ventanas independientes dentro del entorno de simulación.

- **Simulation Control**: este plugin permite controlar la simulación pudiendo pausarla o reiniciarla en cualquier momento. Además, nos permite ejecutar la simulación paso a paso por si estamos realizando tareas de depuración, así como escoger el límite de velocidad que tendrá la simulación.

- **Network**: se utiliza para ver una representación de la distribución de los nodos, pudiendo mostrar ciertas características de todos ellos, como el tipo al que pertenecen, el rango de cobertura e interferencia, el estado de los LEDs, el tráfico generado entre los nodos, etc.
- **Mote Output**: muestra los mensajes de salida de cada uno de los nodos a lo largo de la simulación. Estos mensajes pueden muestran la información procedente de los printf que hay por las diferentes partes del código, mostrando por tanto cuando se reciben los mensajes o notificaciones que vienen de la pila de protocolos.
- **Timeline**: representa una línea temporal para cada uno de los nodos que forman parte de la simulación pudiendo mostrar diferentes eventos como el estado del transceptor radio, el estado de los LEDs u otros eventos relacionados con la transmisión o recepción de los paquetes.
- **Radio Messages**: permite mostrar los paquetes que se transmiten por el medio radioeléctrico con toda su estructura de cabeceras. En él se recogen los mensajes de señalización, de control o de datos de los diferentes protocolos que intervienen en la comunicación como IEEE 802.15.4, ICMPv6, etc. Esto nos puede servir ya que es posible exportar estos datos para analizarlos con el software Wireshark, que nos permitirá identificar los distintos tipos de cabeceras de cada protocolo.
- **Simulation Script Editor**: Cooja permite realizar pequeños scripts desarrollados en Javascript que permiten controlar determinados aspectos de la simulación, como poder automatizar la simulación o variar características del medio radio.
- **DGRM Links Configurator**: este plugin solo aparece en el modo de propagación DGRM y permite configurar los diferentes enlaces de la red de manera asimétrica, pudiendo cambiar el porcentaje de paquetes que se reciben correctamente, el valor de RSSI, el parámetro LQI o el Delay.

En la Figura 42 se puede ver el entorno de simulación de Cooja, mostrando algunos de las ventanas comentadas anteriormente. Esta simulación se corresponde con el ejemplo de transmisión de mensajes broadcast, y se puede ver como los nodos establecen enlaces entre ellos y como se muestran los paquetes recibidos en la ventana de Mote Output.

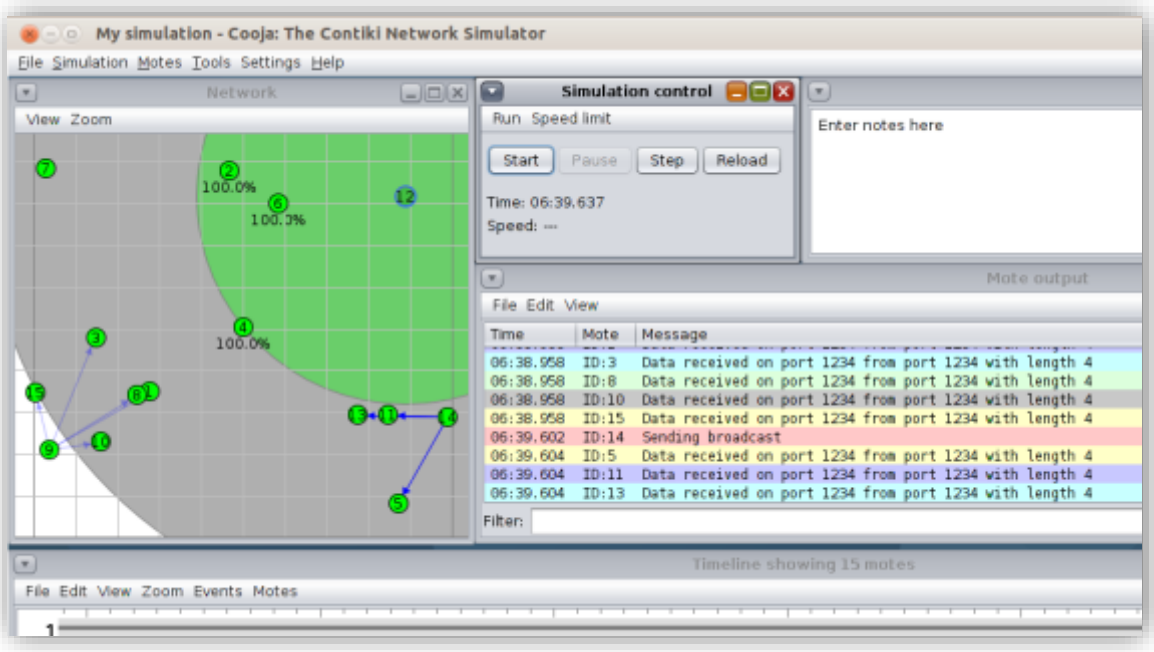


Figura 42. Simulación de una WSN emitiendo mensajes broadcast.

4.2 Selección de rutas de manera eficiente energéticamente

Uno de los objetivos en los que se ha centrado el grupo de investigación recientemente, ha sido el de poder definir un tipo de herramienta o mecanismo que consiguiera facilitar la tarea de desplegar una WSN, sin la necesidad de recurrir a herramientas más complejas como simuladores de campo o planificadores de redes que prolongaban y dificultaban la tarea del despliegue, teniendo en cuenta además, que en la mayoría de situaciones los resultados que se consiguen de simuladores difieren de los que se obtienen una vez la red está desplegada [36] [37], por lo que siempre es necesario realizar una etapa posterior en la que se ajusta la posición de los diferentes nodos.

Es importante considerar que las diferentes soluciones de WSN no suelen desplegarse por expertos en comunicaciones, por lo que esta complejidad que presenta su despliegue se ve aún más acusada para este tipo de usuarios. La fase de despliegue normalmente requiere una planificación previa, estudios de cobertura, pruebas de conectividad, y todo ello se vuelve aún más complejo cuando el sistema se tiene que estar continuamente instalándose y desinstalándose, como podría ser en tareas de inspección o de auditorías. Las soluciones de WSN deben ser capaces de formar una red automáticamente que siga funcionando de manera autónoma, para que el instalador pueda dejar las instalaciones sabiendo que la red funcionará de manera estable. Durante el tiempo de operación, el sistema deberá ser capaz de auto repararse, y ser capaz de cambiar su planificación en el caso de evaluar enlaces con una calidad de conexión pobre o interferencias en los canales. La vida de las baterías también es una característica clave para conseguir aumentar el tiempo en el que la red está funcionando, por lo que resulta interesante implementar algún tipo de mecanismo que permita gestionar los enlaces de forma que consiga aumentar la vida útil de la red. Tanto la complejidad como la vida de las baterías tienen un impacto directo en el coste total del despliegue de este tipo de sistemas.

La solución planteada por el grupo de investigación, llamada Deploy&Forget, ofrece una solución parcial a estos problemas que acabamos de comentar. La solución podrá ser utilizada por operarios durante la fase de despliegue, para ayudar a escoger la mejor ubicación donde colocar los nodos y estará diseñada para trabajar de manera fiable y de forma desatendida maximizando el tiempo de vida de la red. A continuación, se describen las dos características principales de este sistema. En cuanto a la parte de soporte a la fase de despliegue se detalla brevemente su funcionamiento y características principales y en cuanto al funcionamiento desatendido se mostrará las funciones realizadas para conseguir alargar la vida útil de las baterías. Este sistema ha sido patentado por el grupo de investigación [38] y se ha publicado un artículo recientemente en una revista mostrando sus principales características [39].

4.2.1 Asistencia durante la etapa de despliegue

Para poder realizar el despliegue de una red de manera rápida, sencilla y más eficiente será necesario evitar utilizar todas aquellas herramientas [40] [41] o guías adicionales [42] [43] [44] que añaden una complejidad adicional al despliegue en sí mismo. Esto implica que el propio nodo que está siendo desplegado necesita de algún tipo de mecanismo o herramienta para poder indicar al encargado de desplegar la red si se tiene conectividad con la red y con qué calidad de enlace se llega hasta el coordinador. De esta forma, el nodo necesitará de un indicador mediante LEDs, pantallas o cualquier otro sistema que advierta al usuario de esta información. Además, será necesario desarrollar un software adicional que será el encargado de realizar los diferentes test de conectividad y calidad utilizando información de la red a la que pretende unirse. En la Tabla 4 se describe el proceso que seguiría un operario a la hora de desplegar los diferentes nodos que forman la red, ubicando cada uno de ellos de manera iterativa en las localizaciones donde resulta interesante recoger la información mediante los sensores.

Este mecanismo resulta interesante por el hecho de que es el propio nodo el que se encarga de aconsejar cual es el mejor lugar donde colocarlo, para establecer un enlace de calidad con el coordinador a través de la red. Además, este cálculo del nivel de calidad del enlace se realiza a

lo largo de todos los nodos que están involucrados en la comunicación hasta el coordinador, por lo que sería un valor de calidad acumulado en lugar de medir la calidad del enlace de un único salto, como podría ser el valor de RSSI que se tiene con el siguiente nodo. En escenarios como en los que se están planteando, en los que las WSN forman una estructura mallada, la disponibilidad del enlace de comunicación dependerá no solo del enlace de comunicación con el siguiente nodo, si no de todos los que formen parte del camino hasta llegar al coordinador.

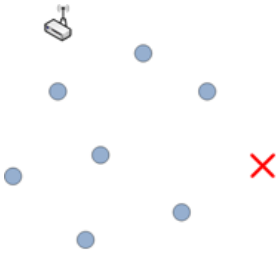
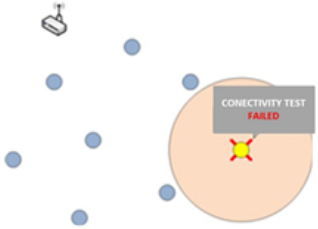
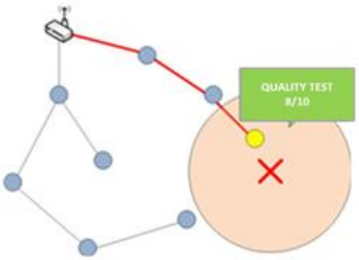
	1. El usuario está en proceso de desplegar la red. El siguiente punto donde es necesario emplazar un nodo para que recoja información es el que está marcado en rojo.
	2. El usuario decide desplegar el nodo en el centro del área donde necesita recoger los datos. Realizando el test de calidad, el usuario revisa si el dispositivo tiene conectividad y cuál es la calidad de la conexión hasta el coordinador de la WSN. Una vez realizado el test de conectividad, el usuario se da cuenta de que en esa posición no se consigue establecer conexión con ningún nodo de la red.
	3. El usuario mueve a continuación el nodo a una posición más cercana a otro nodo y vuelve a realizar el test. En este caso la conexión se ha establecido con éxito, pero la calidad que devuelve el test es muy baja. Esto puede deberse a diferentes factores como el número de saltos, la latencia, la perdida de paquetes o el nivel de señal a ruido, resultando en una conexión no muy fiable. La información sobre la calidad se proporciona al usuario de la manera más simple posible para reducir así la complejidad en el hardware y para que pueda ser utilizado por personal no experto.
	4. Debido a la baja calidad, el usuario decide realizar otro test de calidad en la nueva posición seleccionada, conectándose a ser posible con otro de los nodos. En este caso el resultado del test muestra un valor de calidad mejor que el obtenido en la posición anterior, por lo que el usuario puede desplegar el nodo y continuar con el siguiente sensor.

Tabla 4. Ejemplo del proceso de despliegue.

El cálculo de la variable de calidad se realiza de manera iterativa partiendo del nodo coordinador. En el momento en que un nodo desee conectarse a la red, este enviará un mensaje solicitando la información a los nodos que estén dentro de su rango de cobertura. Partiendo desde el coordinador, en cada salto los nodos irán añadiendo los datos necesarios para calcular el valor de calidad, obteniendo así una variable acumulada a lo largo del camino. Finalmente, el nodo que se está desplegando recibirá el valor de calidad que podrá ir actualizándose por todos los nodos que forman parte del enlace. De esta forma se puede transmitir información sobre los enlaces que forman el camino en una única variable.

Algunas de las métricas que pueden incluirse para obtener la variable de calidad son:

- **ETX (Expected Transmission Count)** que representa una estimación de la pérdida de paquetes, reflejando el número de retransmisiones necesarias para que el paquete se transmita correctamente. Se trata de una variable que representa a un enlace de un único salto.
- **RSSI (Received Signal Strength Indicator)** que indica el nivel de señal con el que se reciben los paquetes de un vecino.
- **LQI (Link Quality Indicator)** que se utiliza para indicar lo fuerte o fiable que es un enlace de comunicación entre dos nodos.
- **Número de vecinos** que cada uno de los nodos tiene dentro de su rango de cobertura. Esto puede servir para conocer los posibles caminos alternativos que tiene un nodo, en caso de que el camino que tenga seleccionado pudiera fallar.
- **RTT (Round Trip Time)** que representa el tiempo de ida y vuelta de un mensaje desde su origen hasta su destino.

El valor que proporcionará este resultado de calidad en el momento en que se esté desplegando el nodo, por lo que la situación en la que se encuentra el nodo durante el tiempo de ejecución de la red podría cambiar debido a la naturaleza del medio de transmisión o por cambios que se produzcan en otros nodos de la red, cambiando así este valor de calidad que se consiguió en una primera etapa. Sin embargo, estos valores que se utilizan para calcular el parámetro de calidad no cambiarán significativamente con el tiempo, excepto que se produzca algún evento que produzca cambios directos sobre la red.

En el momento en que el nodo ya haya sido desplegado en una localización optima, los diferentes mecanismos de encaminamiento y control de la topología propios de la red serán los encargados de mantener las comunicaciones durante el funcionamiento normal de la red. En el siguiente punto se describirá como la propia red puede gestionar sus caminos para encaminar los datos de forma que se mejore su funcionamiento en términos de eficiencia energética, consiguiendo así aumentar el tiempo de vida de la red.

4.2.2 Modo de operación en estado estacionario

Una vez se hayan desplegado todos los nodos pertenecientes a la WSN, la pila de protocolos de comunicación que se haya escogido será la encargada de asegurar el correcto funcionamiento durante el tiempo de operación de la WSN. Para hacer esto se han utilizado diferentes protocolos descritos en el Capítulo 2, desde la capa física hasta el nivel de aplicación, de tal forma que se optimice la robustez y fiabilidad requeridas en aplicaciones en escenarios industriales.

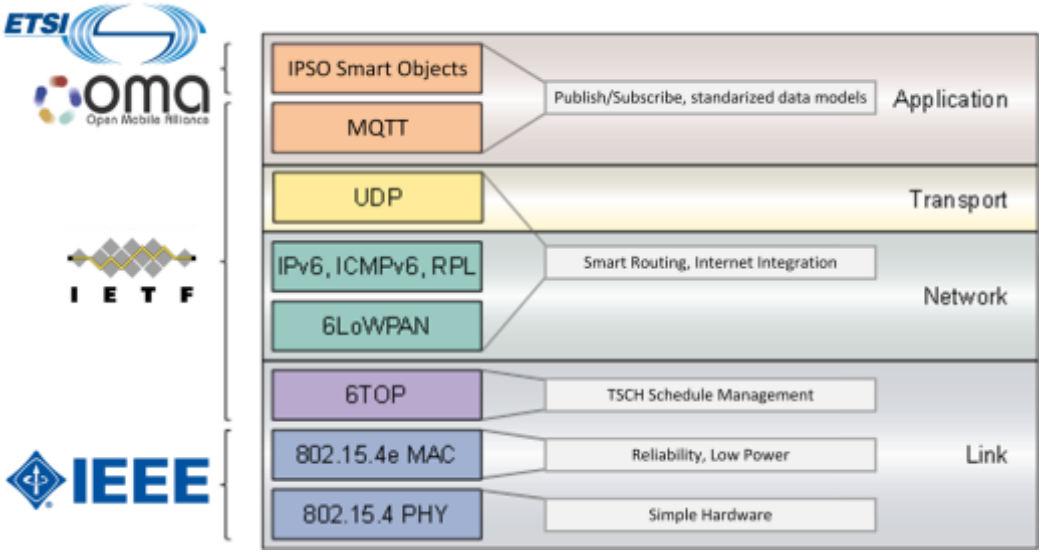


Figura 43. Pila de protocolos utilizada para el despliegue de la WSN.

El proceso sigue el siguiente esquema:

- Identificar la energía restante de un nodo, para una determinada tasa de consumo (esta información será la que intercambiará con el resto de la red para construir la topología).
- Configurar unos límites de batería restante, para poder forzar a los nodos que escojan rutas alternativas cuando un nodo este cerca de acabar su batería.

Este parámetro de energía se podrá intercambiar entre los nodos mediante los mensajes de control (DIO y DIS) del protocolo RPL, para construir la topología y seleccionar el camino más óptimo según las indicaciones de la función objetivo.

Para poder obtener el valor de energía restante de cada uno de los nodos se ha utilizado la herramienta Powertrace [45], que se encuentra implementada en el sistema operativo de Contiki. Powertrace permite llevar un seguimiento del estado de la potencia para poder realizar una estimación de la potencia consumida por el sistema, obteniendo una estructura de datos llamada "energy capsules" para atribuir consumos de energía a diferentes actividades como la transmisión o la recepción de un paquete. De esta forma se puede dividir la potencia consumida por un sistema en actividades individuales que nos permiten conocer cuál de los procesos es el que más afecta al consumo de energía.

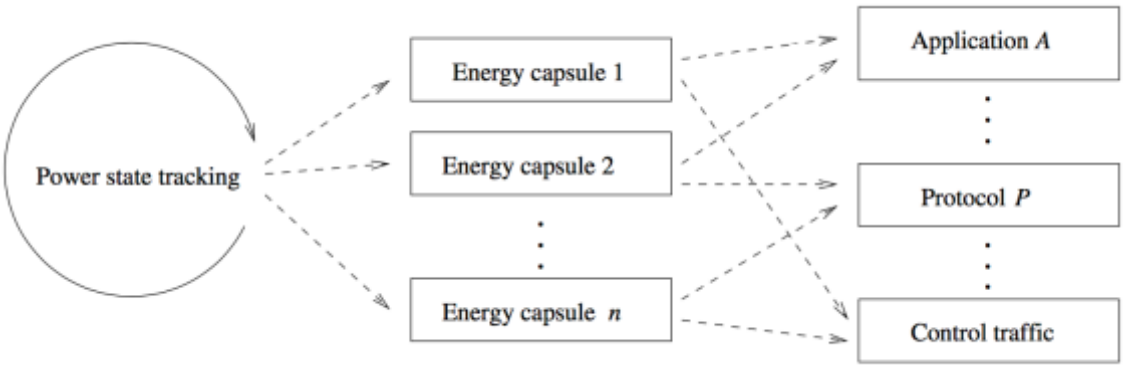


Figura 44. Esquema de análisis del consumo de energía de diferentes actividades.

Esta herramienta está implementada en Contiki por lo que es posible añadirla a la función objetivo de RPL para realizar las estimaciones de energía necesarias. La función de estimación de energía establece un contador que registra la cantidad de ticks de procesador durante el que un determinado momento ha estado activamente consumiendo energía, como por ejemplo el transceptor radio en el modo de transmisión o recepción (cada modo tienen un valor de consumo de energía diferente), el consumo de la CPU del sistema embebido, el tiempo que el sistema entra en el modo de bajo consumo o Low Power Mode (LPM), así como otros elementos presentes como LEDs, puertos entrada salida, etc.

Conociendo la cantidad de tiempo que está activo, la corriente consumida y el voltaje que se suministra para cada actividad, se puede estimar el valor de energía total consumida y por lo tanto la energía restante del dispositivo.

$$p \rightarrow energy_{estimation} = \frac{Energy\_est\_value_i \cdot Current_i \cdot Voltage}{RTIMER_{second} \cdot Runtime}$$

$$i = (TX, RX, CPU, LPM, \dots) \quad \textbf{(4.1)}$$

$$RTIMER_{second} \approx 32768\ ticks/second$$

Esta implementación tiene en cuenta la transformación de la magnitud de tiempo en segundos a partir de la cantidad de ticks del procesador. El resultado de la Ecuación 4.1 proporciona un valor en mW por segundo, indicando lo rápido que el nodo está consumiendo su batería.

La intención de implementar este modo de operación para escoger las rutas de encaminamiento, es el de obtener una métrica que no solo obtenga una métrica que represente la cantidad restante de las baterías en términos de mWh, si no que también permita identificar lo rápido que se está descargando. En un momento dado, un nodo que tenga menos batería restante, pero con un consumo de energía más lento podrá estar funcionando durante un periodo más largo que otros nodos cuyo consumo de batería sea más acelerado.

Algoritmo 4.1 wsn_nm_best_parent(NM_node p1, NM_node p2)

```
metric1 = wsn_nm_path_cost(p1)
metric2 = wsn_nm_path_cost(p2)

/* Un valor de métrica mayor es peor */

mínimum_diff = ENER_SWITCH_THRESHOLD
if p1 is best_parent then
      if metric2 > metric1 + minimum_diff then
            return best_parent = p1
            else return best_parent = p2
      endif
endif

if p2 is best_parent then
      if metric1 > metric2 + minimum_diff then
            return best_parent = p2
            else return best_parent = p1
      endif
endif
```

Algoritmo 4.2 wsn_nm_path_cost(NM_node p)

```
/* Si la energía es 0, significa que la bacteria se ha agotado o que
el nodo es el sink */
/* Si es el sink, la métrica igual a 0 siempre se corresponderá con un
rango menor */

energy_metric = wsn_energy_value()
if energy_metric == 0 and p → rank > sink → rank then
      return cost = WSN_NM_MAX_PATH_COST

      /* Coste infinito cuando la bacteria se haya agotado */
else
      /* Se coge el coste desde el padre hasta el sink (uso de las
      baterías de los padres correspondientes) y se le suma el
      coste del propio nodo */

      link_cost = energy_metric
      return cost = p -> energy_estimation + link_cost
endif
```

La selección de un nuevo padre se puede provocar debido a dos causas diferentes: el nodo detecta algún cambio o irregularidad en la red (por ejemplo, que el nodo pierda la conectividad con su padre o que un nuevo nodo se una a la red), o debido a que uno de los temporizadores especificados expire (el periodo de este temporizador es corto cuando se produce algún cambio en la red, y va incrementando con forme pasa el tiempo sin que se produzca ningún cambio, de esta forma se evita enviar tráfico innecesario, aun así tiene un límite para ser capaz de reaccionar ante posibles cambios).

El código implementado en la función objetivo (OF) de RPL, cuyo pseudocodigo puede verse en los Algoritmos 4.1 y 4.2, incluye un mecanismo de histéresis que permite a los nodos de padre si la diferencia entre las métricas entre el actual y el nuevo candidato es lo suficientemente alta, permitiendo mantener una cierta estabilidad de la red.

4.2.3 Campaña de medidas

Los experimentos realizados para observar el comportamiento de la aplicación desarrollada, se han llevado a cabo en un escenario real al aire libre, pero siempre restringiendo el área de despliegue para poder forzar una topología mallada, lo que nos permitirá analizar el ahorro de energía a diferentes distancias del nodo coordinador. Los dispositivos utilizados han sido los RE-Mote de Zolertia y su interfaz radio CC2538 que se describen en los capítulos anteriores. Estos dispositivos se han programado utilizando el sistema ContikiOS 3.0, junto a una aplicación que envía simples paquetes UDP con información relevante.

Como función objetivo por defecto, Contiki utiliza la tradicional métrica de ETX (Expected transmission Count), que permite seleccionar entre los diferentes caminos con mejor calidad, minimizando la perdida de paquetes. Siendo N_X el número de paquetes recibidos por un nodo X, la métrica ETX entre dos nodos i y j se calcula de la siguiente forma:

$$ETX_{ij} = \frac{N_i}{N_j} \quad \textbf{(4.2)}$$

En este caso, se utiliza como función objetivo lo descrito en [46] (The Minimum Rank with Hysteresis Objective Function, o MRHOF), habilitando la opción que permite utilizar como métrica el parámetro ETX al cuadrado, lo que favorecerá que se escojan buenos enlaces sobre los caminos más cortos, tal y como se recomienda en la RFC. Durante los experimentos, se compararán las dos funciones objetivo para ser capaces de conocer cómo se comportan cada una de manera individual, pudiendo configurar fácilmente una u otra dependiendo del interés de la aplicación que se vaya a desplegar.

En la Figura 45 se puede ver la estructura de la red desplegada, donde se ha utilizado una red mallada de 10 nodos (9 nodos más el coordinador de la WSN). La topología física se ha escogido de tal forma que se forzase una topología lógica que presentará comunicaciones multi-salto, por lo que algunos nodos no se podrán conectar directamente con el sink. Para esto se realizaron diferentes pruebas de cobertura para validar el alcance de los nodos, utilizando diferentes valores de potencia de transmisión.

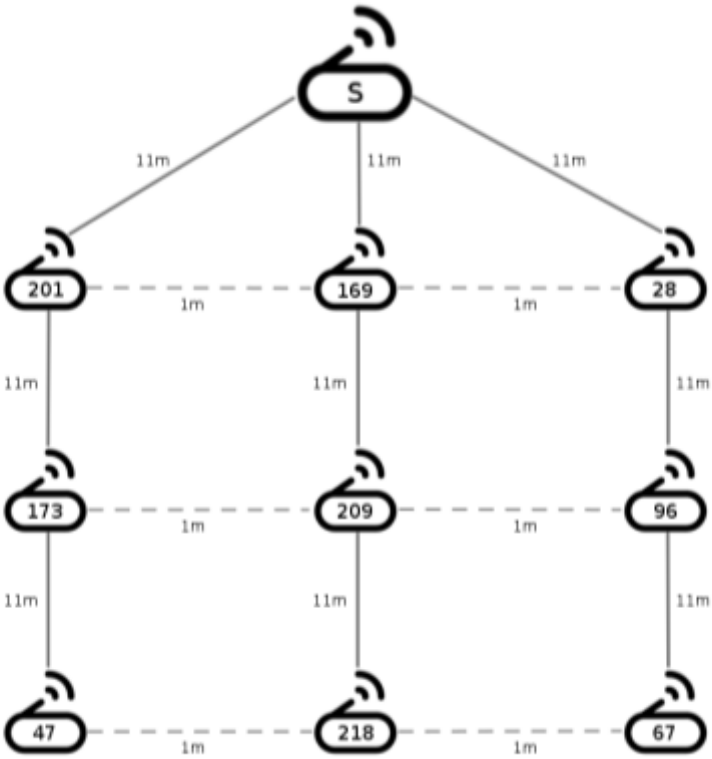


Figura 45. Despliegue de la red durante la campaña de medidas.

El resto de configuración relativa al consumo hardware y de la pila de protocolos se pueden ver en la Tabla 5 y la Tabla 6, donde se han escogido los parámetros que presentaban un mejor rendimiento de la red, minimizando así la pérdida de paquetes y los problemas de sincronización.

Parámetros	***Valor***
Tamaño slotframe TSCH	11 slots
Número de canales	4
Slots activos	4 slots
Canales utilizados	15, 20, 25, 26
Periodo generación de datos	5 segundos
Potencia de transmisión	0 dBm
Threshold ETX	1.25
Threshold Energía	Consumo un 1% más rapido

Tabla 5. Configuración de parámetros de comunicación y TSCH.

Elemento Hardware	*Consumo corriente*
Radio RX	24 mA
Radio TX	20 mA
CPU en LPM	7 mA
CPU	1.3 mA

Tabla 6. Diferentes consumos de corriente de los elementos hardware.

En la Figura 46 se puede ver una imagen del despliegue realizado para llevar a cabo las pruebas, colocando los distintos niveles con una separación de 11 metros.

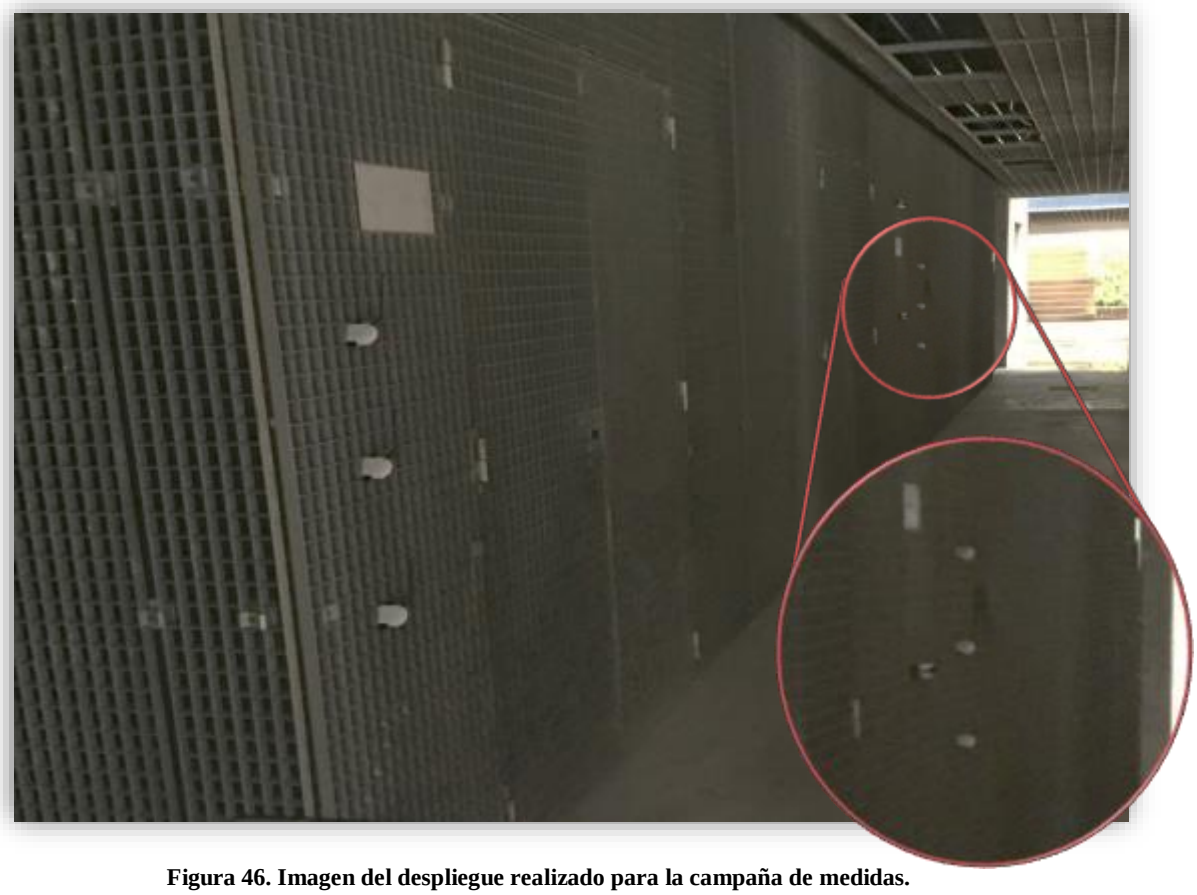


Figura 46. Imagen del despliegue realizado para la campaña de medidas.

4.2.4 Resultados

En este apartado se analizarán los resultados de las pruebas llevadas a cabo para comparar las funciones objetivo de ETX y de Energía. La Figura 47 muestra los resultados en términos de tiempo de vida de las baterías para los nodos de cada nivel. Se puede observar como la función objetivo de energía consigue un mayor tiempo de vida que para el caso de ETX. Durante los diferentes experimentos realizados, se ha conseguido que los nodos del primer nivel estén operativos durante un 15% más de tiempo, y hasta un 22% para los nodos que formaban parte del grupo más alejado del coordinador. Como se esperaba, los nodos que se encuentran más cercanos al coordinador consumen sus baterías más rápidamente debido a que necesitan transmitir más paquetes que los otros nodos.

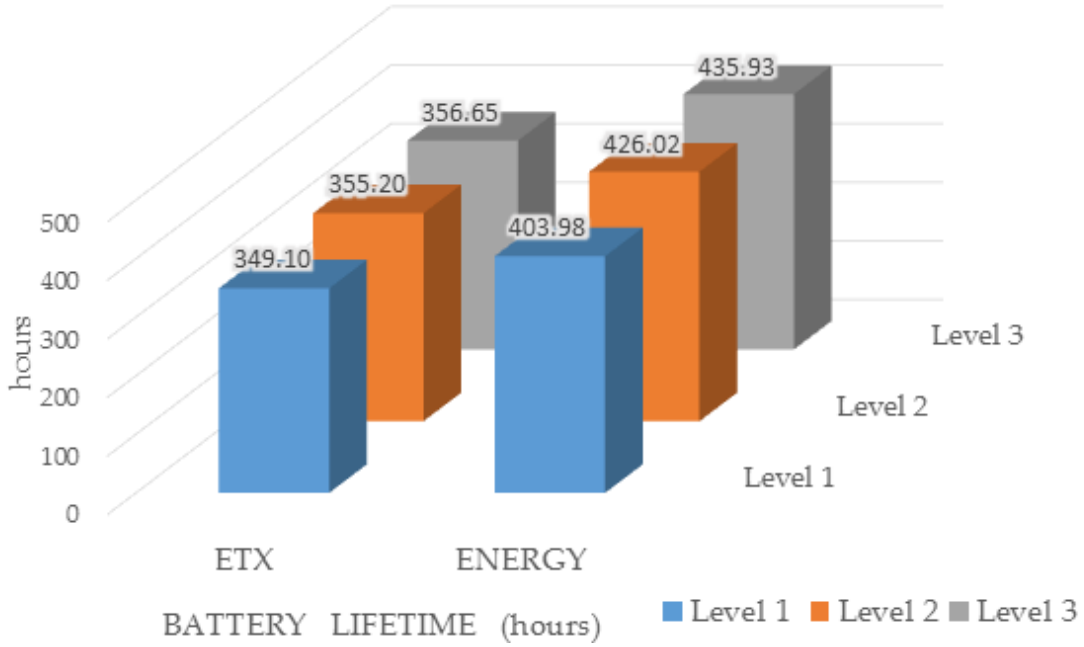


Figura 47. Comparación del tiempo de vida de las baterías conseguido durante los experimentos.

En las siguientes figuras se podrán ver detalles más en profundidad que nos permitirán analizar el modo de operación de la red. En la Figura 49 y Figura 50 podemos ver la tendencia que sigue el consumo de energía del último nodo en agotar su batería de los del primer nivel, mostrando desde el instante 2 horas antes de que el primer nodo consuma su batería hasta que todos los nodos del primer nivel se desconectan de la red, comparando la tendencia de las dos configuraciones, ETX y energía. En la Figura 48 se puede ver en más detalle que elementos del hardware son los que están provocando el consumo de energía (transmisión, recepción y el total en mW), agrupando los nodos por nivel. De esta gráfica podemos deducir que, aunque la función objetivo de energía muestra un mayor consumo debido al transceptor radio en transmisión, sí que se reduce el consumo cuando este está recibiendo. Mediante esta O.F de energía se consigue que el tráfico esté balanceado entre los nodos para intentar que las baterías se descarguen más lentamente, mientras que con ETX los nodos mantienen su tráfico continuo hasta que las baterías se agotan.

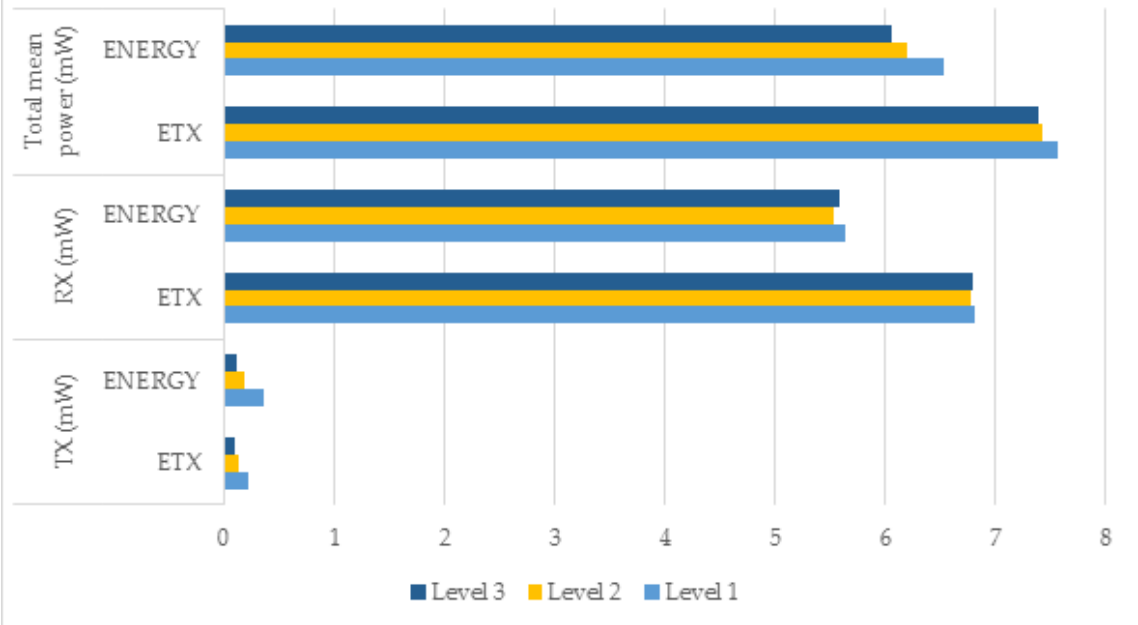


Figura 48. Perfiles de consumo de potencia relevantes medidos durante las pruebas.

En la Figura 49 y Figura 50 se puede ver el comportamiento de uno de los nodos cuando el resto de dispositivos del primer nivel se desconectan. En el caso de ETX, los nodos comienzan a agotar sus baterías mucho antes, lo que provoca que el resto de nodos tengan que soportar la carga de tráfico del nodo que ha dejado la red, lo que se refleja en un mayor consumo por los nodos restantes. Por otro lado, la O.F. de energía permite que estos nodos agoten sus baterías mucho más tarde (alrededor de 61 horas de diferencia entre el instante en que los primeros nodos dejan la red), consiguiendo balancear la potencia consumida entre los vecinos de la red.
El balanceo de las baterías de los nodos tiene además otro beneficio, y es que al mantener un mayor número de vecinos disponibles durante un mayor tiempo se incrementa la flexibilidad de la red, ya que se tienen más caminos disponibles por los que encaminar los datos.

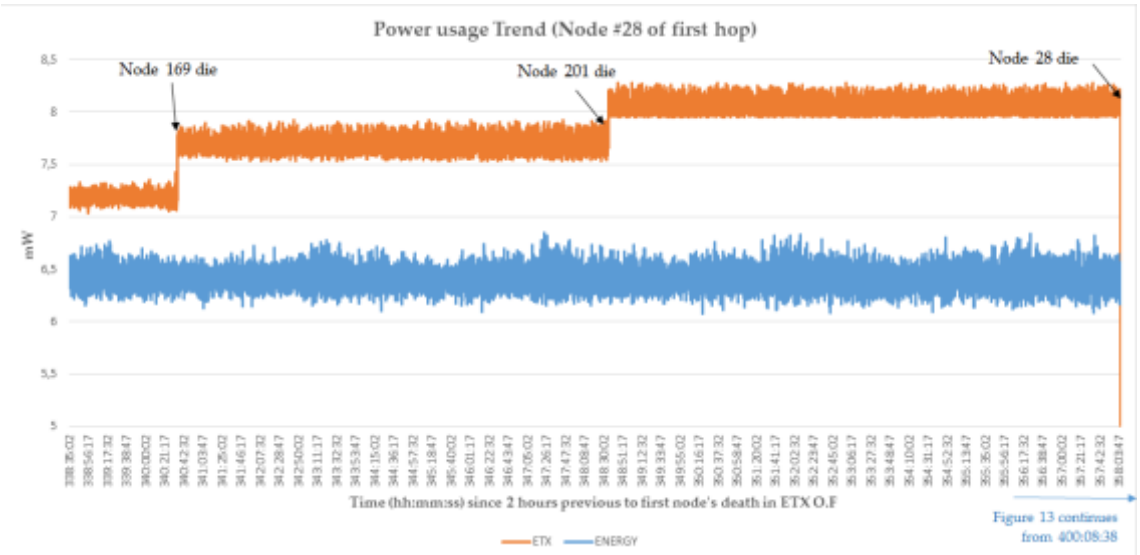


Figura 49. Tendencia de consumo de potencia del último nodo en consumir la batería del primer salto, mostrando desde 2 horas antes de que la primera batería de los nodos se agote, hasta que todos los nodos del primer nivel mueren en el experimento con la O.F. de ETX

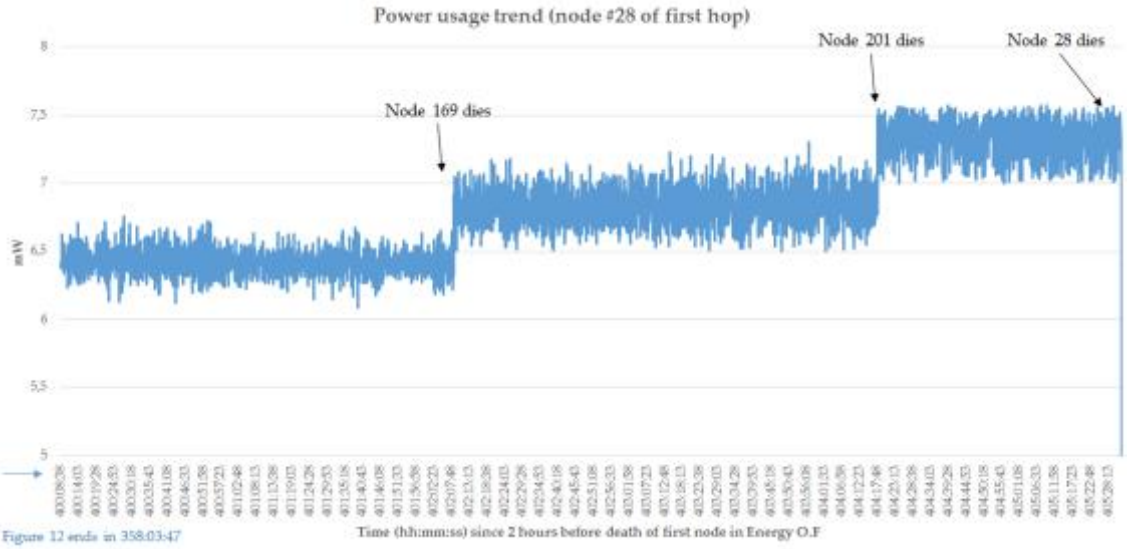


Figura 50. Continuación de la Figura 49, mostrando la tendencia del consumo de potencia del último nodo en consumir su batería del primer nivel, mostrando desde 2 horas antes de que la primera batería de los nodos se agote, para el experimento con la O.F. de Energía, mostrando que esto ocurre varias horas después que para el caso de ETX.

4.3 Estimación de calidad

Para esta segunda fase del proyecto, se va a continuar desarrollando en más profundidad lo presentado en el apartado anterior sobre el mecanismo de despliegue rápido. Como ya se ha mencionado, el despliegue de las WSN es uno de los inconvenientes que está ralentizando su introducción en la industria, debido al tiempo que puede llevar realizar un despliegue completo y óptimo, además de que puede resultar ser un desafío para aquellas personas que no tengan los conocimientos necesarios sobre comunicaciones y calidad de los enlaces. Anteriormente se ha definido un mecanismo, denominado Deploy&Forget que pretendía facilitar estas tareas de despliegue, evitando utilizar otras herramientas o simuladores adicionales. Este mecanismo conseguía indicar la ubicación optima donde colocar el nodo, en función de la conectividad con la red y de la calidad del camino hasta llegar al coordinador. Este valor de calidad puede ser utilizado también para ayudar al protocolo RPL para que tome las decisiones necesarias para construir una topología con enlaces fiables, basándose en la calidad acumulada que presenta el camino completo entre origen y destino. El objetivo de esta parte del proyecto es definir un estimador de calidad que permita caracterizar los enlaces y caminos entre los diferentes nodos que forman la red durante las primeras etapas de despliegue, permitiendo al protocolo RPL construir una topología robusta utilizando enlaces fiables. Los resultados de estas pruebas se han recogido en un artículo que se presentará al congreso TELFOR que se llevará a cabo en Noviembre de 2017, se incluye su referencia aunque su estado está en proceso de aceptación [47].

Este parámetro de calidad estará construido a partir de la combinación de diferentes métricas que caracterizan la calidad del enlace: el nivel de señal de los paquetes recibidos representado mediante el parámetro RSSI (Received Signal Strength Indicator); el parámetro ETX (Expected Transmission Count) que representa el número de transmisiones necesarias para que un paquete se transmita correctamente, lo que se traduciría a la cantidad de pérdidas que se producen al transmitir información por un enlace; por último el número de saltos que hay desde un nodo hasta el coordinador de la red, que se puede interpretar como un parámetro aproximado de la latencia que hay entre un nodo y el root.

Esta aproximación de la latencia utilizando el número de saltos se realiza debido a que una implementación exacta que permita evaluar dicha métrica requiere de un intercambio adicional de información que permita tener una base de tiempos con la que marcar los diferentes paquetes durante el establecimiento de la comunicación. Este intercambio extra de paquetes es algo que

hemos querido evitar, utilizando una aproximación más sencilla con el número de saltos. La justificación de esta aproximación se basa en la utilización de un mecanismo de acceso al medio determinista como es TSCH, con una tasa de generación de paquetes que es mucho más alta que la duración de un slotframe de capa MAC. Cada paquete necesitará un slot por cada salto para ser transmitido, por lo que se tendrá suficiente margen para reenviar cualquier paquete que haya en la cola antes de acabar un periodo de transmisión de mensajes. Por lo tanto, se ha asumido que la latencia end-to-end se corresponde con la duración de un frame por cada salto hasta llegar al coordinador.

Una vez definidas las variables que serán utilizadas para estimar la calidad, los pasos para llevar a cabo el cálculo de la función de calidad serán los siguientes:

- Determinar el valor de las métricas que intervendrán en la función de calidad, almacenando dichos valores de manera organizada y actualizando su valor siempre que sea posible.
- Crear una estructura de datos donde tener almacenadas cada una de las métricas para cada uno de los vecinos dentro del rango de cobertura, y en función del canal de comunicaciones siempre que sea posible.
- Una vez los datos estén disponibles, se utilizarán unas funciones de mapeo para homogeneizar los niveles de cada uno de los parámetros. Los resultados de estas funciones de mapeo serán las que formen el valor de calidad del enlace, dando un peso diferente a cada una de las métricas que podrá modificarse en función de los requisitos de la aplicación.
- Definir unos límites de calidad que serán los que determinen que camino se debe escoger, en función de la diferencia de calidad que presenten los diferentes caminos disponibles.

La implementación de este método de estimación de la calidad del enlace se llevará a cabo sobre la pila de protocolos del sistema Contiki, de esta forma se podrá utilizar para diferentes propósitos como servir a RPL para construir su topología o para mostrar a través del hardware de los RE-Mote una representación visual del parámetro mediante LEDs, pantallas u otro tipo de periféricos que permitan visualizar de manera sencilla si la ubicación donde se va a emplazar el nodo tiene suficiente calidad.

Ya que realizamos la implementación en el sistema Contiki, tendremos la ventaja de que este sistema ya implementa un mecanismo para evaluar el valor de ETX de los enlaces que tiene activos un nodo con el resto de sus vecinos. Contiki dispone de una serie de funciones que permiten al sistema evaluar diferentes tipos de estadísticas sobre los enlaces que utiliza un nodo, entre estos está el parámetro ETX. Una de estas funciones permite obtener el número de transmisiones realizadas para enviar un paquete, así como una variable de estado que indica si esta transmisión se ha realizado con éxito. De esta forma, conociendo el número de transmisiones necesarias para que la transmisión se realice con éxito, podremos estimar el ETX para cada uno de los vecinos con los que se esté intercambiando información. Es importante resaltar que, debido a que es necesario conocer si el paquete se ha transmitido con éxito, no todos los mensajes se podrán utilizar para estimar el valor de ETX, como es el caso de las Beacons que no requieren de ACK de reconocimiento. Por lo tanto, los únicos mensajes que contribuirán al cálculo de esta métrica serán aquellos mensajes de datos cuya transmisión sea unicast y también los mensajes de control de RPL transmitidos también en unicast, ya que también utilizan los ACK.

De esta forma los dispositivos tendrán una estructura donde almacenarán los valores de ETX para cada uno de los vecinos, incluyendo su padre con el que intercambia directamente información. Los valores de este parámetro se irán actualizando cada vez que se realice una transmisión con alguno de los vecinos, lo que hace que para aquellos vecinos con los que se intercambie mensajes frecuentemente, tendrán un parámetro de ETX más actualizado que para aquellos con los que solo intercambie mensajes de control para mantener información sobre la

topología. Esto puede generar que los cambios de padre sean algo lentos debido a que el ETX del padre actual va cambiando con más frecuencia que los posibles candidatos.

Estas funciones que permiten obtener el parámetro ETX también realizan un filtrado de las muestras de tipo EWMA (Exponential Weighted Moving Average) como el que se muestra en la Ecuación 4.3. De esta forma se consideran valores medidos en instantes anteriores para mejorar la estabilidad de la red, ya que si en un momento puntual el ETX se disparara por un momento se forzarían cambios de padre de forma continua, pudiendo provocar que se llegaran a perder mensajes. En Contiki este filtrado se realiza con un factor $\alpha = 0{,}15$ lo que daría un peso del 85% a los valores medidos en instantes anteriores. Estas funciones tienen además implementadas un mecanismo de temporización que permite identificar cuando un valor no está lo suficientemente reciento, expirando un temporizador que iniciaría un proceso para actualizar dicho valor. De esta forma se controla que los datos sean recientes, pudiendo modificar los pesos del filtrado EWMA con un factor de $\alpha = 0{,}3$ para que los parámetros cambien más rápidamente hasta su valor estable.

$$ETX = \alpha \cdot etx_t + (1 - \alpha) \cdot etx_{t-1} \quad \textbf{(4.3)}$$

En cuanto a la obtención del parámetro RSSI, ha sido necesario desarrollar una serie de funciones que permitieran evaluar dicho parámetro para cada canal en el que se estén estableciendo las comunicaciones y para cada uno de los vecinos dentro del rango de cobertura de un nodo. Estas funciones se han incluido al nivel de acceso al medio, para permitir que procesen todos los mensajes que reciba un nodo, pudiendo obtener de los paquetes información como las direcciones de origen y destino, el canal de frecuencia en el que se han establecido las comunicaciones, el nivel de señal o RSSI con el que se ha recibido el mensaje y otros valores como la calidad (LQI). De esta forma, utilizando las direcciones de origen podemos crear una matriz de información que guarde los valores de RSSI para cada uno de los vecinos en función de los canales en los que se han recibido los mensajes. A diferencia de otras soluciones, esta nos permite evaluar el nivel de señal que presenta cada uno de los canales de manera independiente, lo que permite detectar si uno de ellos es el que está perjudicando el intercambio de mensajes debido a interferencias en ese canal, y pudiendo tomar medidas como crear una lista negra con los canales en los que no se desea transmitir datos.

A estos valores de RSSI por canal se le realizará un filtrado de tipo EWMA, de la misma forma que se hace con los valores de ETX, tal y como se muestra en la Ecuación 4.4, con unos valores de α iguales que los utilizados en ETX. Esto nos permite tener una estructura como la que se muestra en la Tabla 7, obteniendo un valor promediado de los valores de RSSI para cada canal que será el que utilicemos en la función de calidad.

$$RSSI^{ch} = \alpha \cdot rssi_t^{ch} + (1 - \alpha) \cdot rssi_{t-1}^{ch} \quad \textbf{(4.4)}$$

	$RSSI^{ch\_1}$	...	$RSSI^{ch\_n}$
Nodo A			
Nodo B			

$RSSI$
$\frac{\sum_{ch=1}^{n} RSSI_{nodo\ A}}{n}$
$\frac{\sum_{ch=1}^{n} RSSI_{nodo\ B}}{n}$

Tabla 7. Tabla de estimación de RSSI por vecino en cada nodo.

El motivo por el que no se obtienen los valores de ETX para cada canal, de la misma forma que hacemos para el RSSI, es debido a la propia definición de este parámetro y del método de acceso al medio utilizado. Este parámetro se obtiene contando el número de retransmisiones necesarias de un paquete, por lo que, si quisiéramos obtener un valor de ETX para un determinado canal, las retransmisiones se deberían realizar únicamente por ese canal. Por lo tanto, al utilizar TSCH como método de acceso al medio, en cada transmisión el dispositivo utilizará un canal diferente para retransmitir los paquetes que no se hayan entregado con éxito.

Finalmente, solo queda obtener el número de saltos que hay desde un nodo hasta llegar al coordinador. Para poder evaluar este parámetro es necesario realizar un intercambio de información que nos permita conocer por cuantos nodos pasamos para llegar hasta el coordinador, ya que el dispositivo únicamente conoce los vecinos que hay dentro de su rango de cobertura, entre los que se incluye su padre, pero no los nodos que hay por encima de estos hasta llegar al coordinador. Sin embargo, nuestra intención es evitar transmitir todo tipo de tráfico adicional por lo que se han utilizado los mensajes de control de RPL como contenedores de una variable que se va incrementando un valor fijo cada vez que pasa por un nodo.

Los mensajes de control DIO son los encargados de transmitir información sobre el rango, que indica la distancia que hay hasta el coordinador teniendo en cuenta las métricas utilizadas. Este rango no se puede utilizar para contar el número de saltos ya que, aunque se va incrementando cada vez que se aleja un salto más del coordinador, su incremento será variable dependiendo de las estadísticas que tenga el enlace. Para poder transmitir una variable que se vaya incrementando un valor fijo, se han utilizado los Metric Container de los mensajes DIO, que nos permiten almacenar una métrica que se enviará junto a otra información relativa al enlace, como es el rango. De esta forma, cada vez que un nodo reciba un mensaje DIO extraerá la variable del número de saltos, la incrementará y el resultado lo transmitirá en su propio mensaje DIO para que otros nodos puedan unirse a la red y conocer el número de saltos que hay hasta llegar al coordinador. En este caso, al ser un valor que representa un estado absoluto, no se necesita realizar un filtrado como el que utilizamos para las otras dos métricas, ya que este valor tiene que indicar exactamente el número de saltos que tiene un nodo. En la Figura 51 se puede ver una representación del mecanismo utilizado para obtener el número de saltos, viendo como esta variable se va incrementando en un valor de 128 con cada salto que se aleja del nodo coordinador.

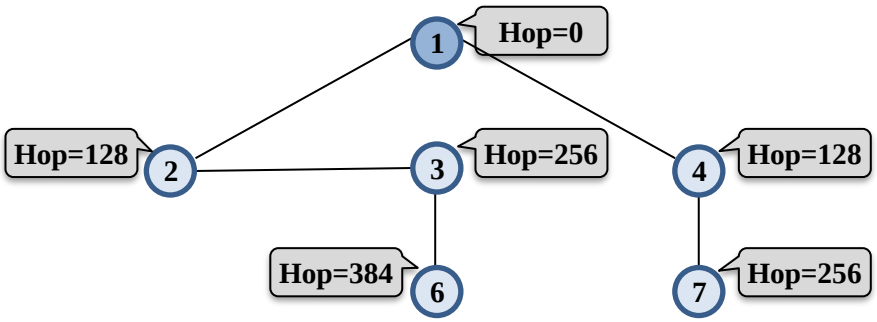
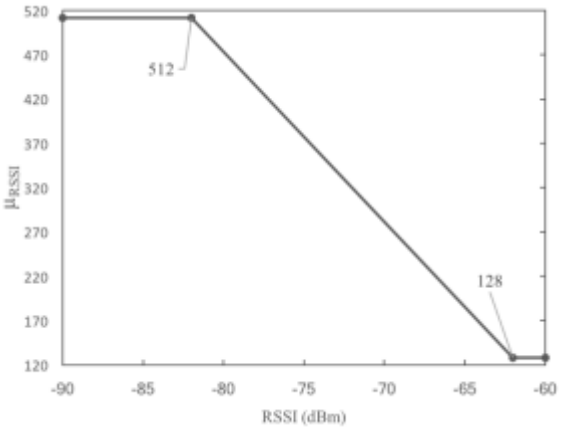


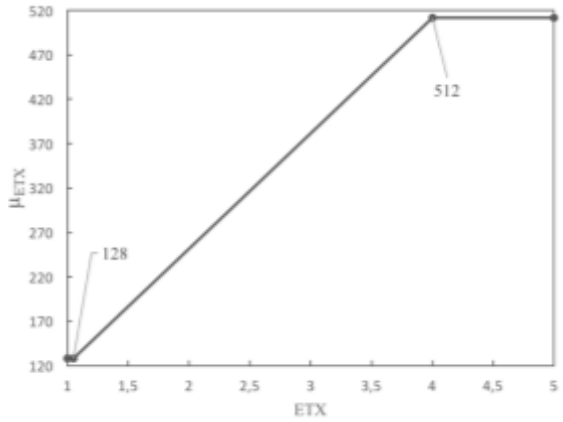
Figura 51. Mecanismo de obtención del número de saltos mediante mensajes de control RPL.

Una vez hemos obtenido las tres métricas que se utilizarán para estimar el valor de calidad, es necesario utilizar unas funciones de mapeo para que todas las métricas se muevan dentro de un mismo rango de valores. En la implementación de la función objetivo MRHOF que realiza Contiki se utiliza un valor máximo de métrica permitido de 1024, que en ETX se correspondería con un valor de 8 ya que está multiplicado por un factor de 128. Sin embargo, en la especificación de la función objetivo [46] no se recomienda utilizar enlaces que tengan un ETX superior a 4 ya que significaría que el 75% de los mensajes no se transmiten correctamente. De esta forma, se han escogido unos límites entre 128 y 512 para implementar las funciones de

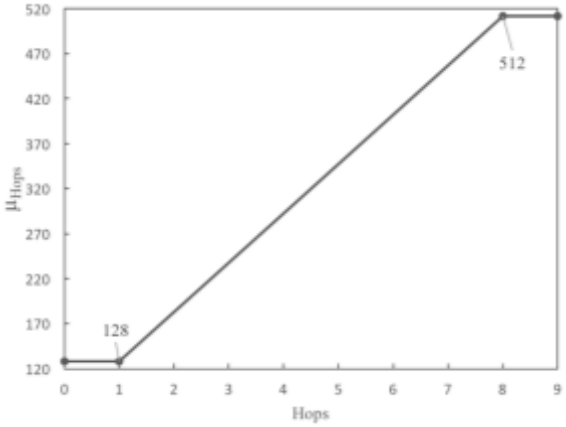
mapeo que se muestran en la Figura 52. En el caso de ETX nos moveremos en un rango de valores entre 1 y 4, por lo que la función de mapeo únicamente deberá multiplicar por el factor de 128. Para el caso del RSSI los límites se han configurado en -62 dBm y -82 dBm para el peor caso y el número de saltos se ha limitado a un máximo de 8 saltos.



(a) Función de mapeo RSSI.



(b) Función de mapeo ETX.



(c) Función de mapeo Hops.

Figura 52. Funciones de mapeo para las diferentes métricas.

Una vez se han mapeado las distintas métricas empleando las funciones descritas en la Figura 52 ya es posible estimar la función de calidad dando unos pesos a cada una de estas métricas, que determinarán su influencia en el resultado final del valor de calidad. En la Función 4.5 se puede ver la ecuación utilizada para obtener el valor de calidad LQS, donde las variables α_i representan los pesos de cada métrica y las variables μ_i son los valores obtenidos de las funciones de mapeo. Los pesos de cada una de las métricas pueden ser fácilmente configurables para obtener distintas variaciones de la función de calidad que darán más importancia a enlaces que presenten mejores características de pérdida de paquetes o de latencia, en función de la aplicación en la que se vaya a utilizar.

$$LQS = \alpha_{RSSI} \cdot \mu_{RSSI} + \alpha_{ETX} \cdot \mu_{ETX} + \alpha_{HOPS} \cdot \mu_{HOPS} \quad \textbf{(4.5)}$$

Finalmente, es necesario implementar un mecanismo de histéresis que mejore la estabilidad de la red, ya que se puede dar el caso en el que dos vecinos presenten un nivel similar de calidad, lo que provocaría realizar continuos cambios de padre por pequeñas que sean las variaciones de estos valores de calidad. De esta manera podemos dar un margen de diferencia entre diferentes métricas a la hora de cambiar de padre para evitar estos cambios continuos. En el Algoritmos

4.3 y el Algoritmo 4.4 se puede ver un pseudocodigo de las funciones utilizadas para realizar el cambio de padre.

Algoritmo 4.3 wsn_qlty_best_parent(node p1, node p2)

```
p1_cost = wsn_path_quality(p1)
p2_cost = wsn_path_quality(p2)

/* Un valor de métrica mayor es peor */

min_diff_etx = ETX_SWITCH_THRESHOLD
min_diff_rssi = RSSI_SWITCH_THRESHOLD
min_diff_hops = HOPS_SWITCH_THRESHOLD

wt_etx = ETX_WEIGHT
wt_rssi = RSSI_WEIGHT
wt_hops = HOPS_WEIGHT
wt_sum = wt_etx + wt_rssi + wt_hops

if p1 is best_parent then
      if p2_cost > ( wt_etx(p1_cost+min_diff_etx) +
                     wt_rssi(p1_cost+min_diff_etx) +
                     wt_hops(p1_cost+min_diff_etx))/wt_sum then
            return best_parent = p1
            else return best_parent = p2
      endif
endif

if p2 is best_parent then
      if p1_cost > ( wt_etx(p2_cost+min_diff_etx) +
                     wt_rssi(p2_cost+min_diff_etx) +
                     wt_hops(p2_cost+min_diff_etx))/wt_sum then
            return best_parent = p2
            else return best_parent = p1
      endif
endif
```

Algoritmo 4.4 wsn_path_quality(node p)

```
/* Obtenemos la calidad del enlace y calculamos la calidad de todo el
camino hasta el sink*/

LQY_metric = wsn_link_quality_estimator()
if LQY_metric == 512 and p → rank > sink → rank then
      return cost = WSN_NM_MAX_PATH_COST

      /* Si no es el sink y tiene un valor de calidad muy bajo (un
      valor alto indica peor calidad) se devuelve como inalcanzable*/
else
      /* Sumamos el coste del enlace al valor de calidad acumulado del
      padre */

      link_cost = LQY_metric
      return cost = p -> quality_estimation + link_cost
endif
```

Una vez tenemos definida e implementada la función de calidad como métrica para utilizar en la función objetivo de RPL podemos pasar a realizar las diferentes simulaciones para comprobar su funcionamiento, en función de los costes de cada métrica que se irán variando en diferentes

configuraciones. A continuación, se detallan las simulaciones que se han realizado y los resultados que se han obtenido de las mismas.

4.3.1 Simulaciones realizadas

Para comprobar el funcionamiento de la función de calidad desarrollada se van a realizar diferentes simulaciones utilizando el simulador Cooja que viene integrado en el sistema Contiki. De esta forma podremos comparar el comportamiento de la función objetivo que utiliza por defecto Contiki, en el que la métrica empleada es el ETX (Expected Transmission Count), con diferentes variaciones de la función de calidad que se ha desarrollado.

Ya que las distintas simulaciones pretenden mostrar el comportamiento de la red frente a variaciones que se puedan producir en los valores de las diferentes métricas utilizas, será importante tener un control sobre ellas para poder modificarlas durante la simulación. De esta forma podremos alterar los valores del porcentaje de pérdida de paquetes o el nivel de RSSI. En Cooja, esta característica que permite modificar las métricas solo la ofrece el medio DGRM (Directed Graph Radio Medium) y su plugin que permite configurar cada uno de los enlaces (DGRM Link Configurator). Este configurador permite cambiar los parámetros de ratio de recepción de paquetes, nivel RSSI del enlace, nivel de calidad LQI o delay. En nuestro caso nos interesará modificar únicamente el ratio de recepción de paquetes y el nivel de señal o RSSI.

Sin embargo, este configurador no permite automatizar el cambio de estos parámetros, por lo que ha sido necesario desarrollar un pequeño programa que lo controle. Cooja implementa un editor de scripts que permiten controlar la mayoría de variables de simulación, como el tiempo de simulación, la posición de los nodos o cambiar parámetros de otros plugin como el DGRM Link Configurator que a nosotros nos interesa.

El script ha sido desarrollado en JavaScript y básicamente implementa un bucle infinito que cada 10 milisegundos (el tiempo de duración de un slot temporal de TSCH) cambiará los valores del ratio de recepción y de RSSI dentro de un cierto rango predefinido. Es importante que este script se ejecute de igual forma que lo haría el slotframe que tenga configurado TSCH, para que los valores vayan cambiando de forma sincronizada con en el script. La configuración de la pila de protocolos es la que se muestra en la Tabla 8. Los valores de RSSI cambiarán de forma aleatoria desde -58 dBm hasta -84 dBm, configurando los enlaces más largos con valores más bajos de RSSI. Para el caso del ratio de recepción los valores variarán desde el 73% de los paquetes hasta el 85%, pretendiendo simular enlaces con pérdidas que afecten al parámetro ETX.

Parámetros	***Valor***
Tamaño slotframe TSCH	7 slots
Número de canales	4
Slots activos	3 slots
Canales utilizados	15, 20, 25, 26
Periodo generación de datos	6 segundos

Tabla 8. Configuración de parámetros de comunicación y TSCH para las simulaciones.

Manteniendo fijos estos parámetros de simulación, se llevarán a cabo 5 tipos de simulaciones que mostrarán diferentes configuraciones de la función objetivo de RPL. Una de estas simulaciones será la utilizada por excelencia, en la que se utilizará simplemente la métrica de

ETX, tal y como se utilizó para las pruebas de energía. Esto nos permitirá comparar los resultados con uno ya conocido como el que implementa la función objetivo MRHOF.

Las simulaciones restantes utilizarán como métrica de la función objetivo, el valor que proporciona la función de calidad que se ha implementado, dando a cada una de las simulaciones un valor a los diferentes pesos que determinan la función de calidad. En 3 de estas simulaciones se configurarán los pesos de tal forma que se utilice una única métrica de las tres disponibles, dando un peso nulo al resto, obteniendo una simulación donde la función de calidad solo utilice el parámetro ETX, otra en la que solo utilice el RSSI y una última en la que solo se emplee el número de saltos como métrica. La simulación restante implementará un valor de calidad dando un peso similar a cada métrica. Estas 5 simulaciones tendrán una duración total de 20 horas cada una, llegando a transmitir algo más de 400.000 paquetes UDP entre todas las simulaciones.

Como tipo de nodos utilizados, se han escogido los nodos Z1 de Zolertia para emular su comportamiento en las simulaciones, ya que el simulador Cooja no disponía de otro tipo de nodos más reciente como los RE-Mote. Cabe destacar que, aunque estos depositos Z1 de Zolertia no tenían la capacidad de utilizar TSCH, ya que implementaban una versión anterior del estándar, en Contiki se han realizado algunas modificaciones para que el software sea capaz de emular esta característica, deshabilitando algunas características que no afectan de ninguna forma a nuestras simulaciones. Escoger el tipo de nodos que deberíamos simular ha sido un problema que afrontar ya que los tipos de nodos que hay implementados en Cooja no son demasiados recientes y todos ellos implementan este pequeño parche para ser capaces de correr TSCH, y debido a las limitaciones de memoria de estos dispositivos ha sido necesario realizar bastantes tareas de depuración para que el simulador no diese problemas al compilar el código para el tipo de nodo escogido.

En cuanto a la topología se ha utilizado una red mallada de 8 nodos (7 nodos más el coordinador de la WSN). Se ha escogido una topología física que permitiera que solo tres de los dispositivos pudieran comunicarse directamente con el sink, forzando así una comunicación multi-salto de al menos 2 saltos en algunos casos. Además, se ha colocado un último nodo en un tercer nivel que necesitará como mínimo de 3 saltos para llegar a su destino, y nos servirá para realizar comparaciones en cuanto al retardo que se produce en las diferentes configuraciones. En la Figura 53 se puede ver una captura de una de las simulaciones en tiempo de ejecución.

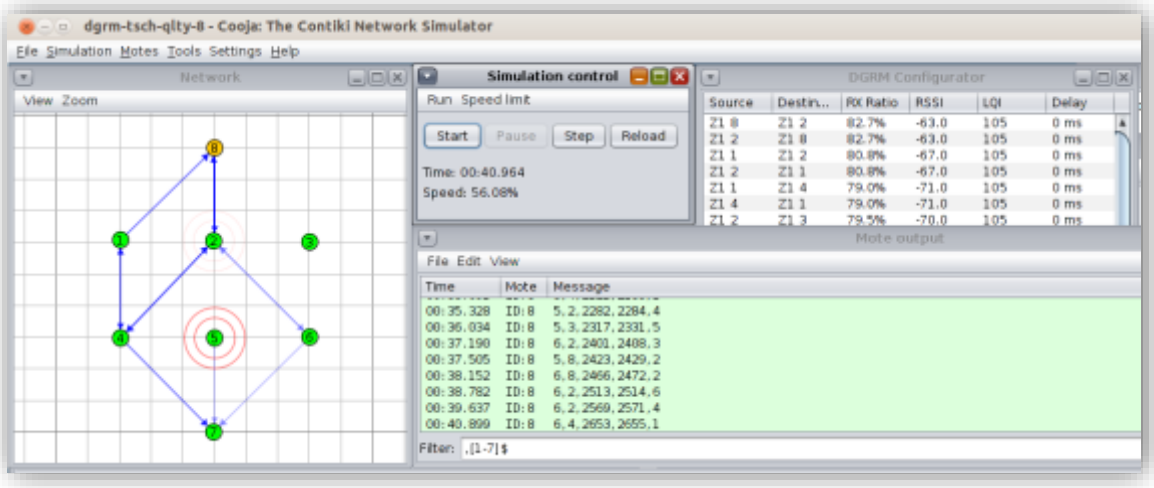


Figura 53. Simulación de la red de 8 nodos utilizando la función de calidad.

4.3.2 Resultados

En este apartado se analizarán los resultados obtenidos en las diferentes simulaciones llevadas a cabo en Cooja. Se van a comparar las 5 configuraciones de funciones objetivo que se han realizado:

- MRHOF: que utiliza la métrica tradicional de ETX que viene implementada en Contiki.
- LQS ($\alpha_{RSSI} = 1$): esta configuración implementa la función de calidad desarrollada dando un peso total a la métrica de RSSI, dejando el resto a cero.
- LQS ($\alpha_{ETX} = 1$): esta configuración implementa la función de calidad desarrollada dando un peso total a la métrica de ETX, dejando el resto a cero.
- LQS ($\alpha_{HOPS} = 1$): esta configuración implementa la función de calidad desarrollada dando un peso total a la métrica del número de saltos, dejando el resto a cero.
- LQS ($\alpha_i = {}^1/_3$): esta configuración implementa la función de calidad desarrollada dando el mismo peso a las tres métricas.

En la Figura 54 están representadas el porcentaje de pérdidas de paquetes que se ha producido durante las simulaciones para cada una de las configuraciones. Estos valores se han podido obtener gracias a que cada mensaje iba identificado mediante un número de secuencia, lo que nos permite conocer si se ha perdido alguno durante todo el tiempo de simulación.

Como se puede apreciar, las configuraciones que presentan un mayor porcentaje de pérdidas son aquellas en las que se utilizan el número de saltos o el nivel de RSSI como métricas para la función de calidad. Esto se debe a que ninguno de estos parámetros tiene en cuenta las pérdidas que se producen en el canal utilizado, intentando escoger aquel canal que minimice dicha métrica sin tener en cuenta si este enlace de comunicaciones provoca más pérdidas que otros.

Los resultados obtenidos para el número de saltos se pueden comparar con la utilización de la Objective Function Zero [RFC 6552] en la que se utiliza la distancia hasta el root como una métrica para construir la topología RPL y escoger entre las diferentes rutas de encaminamiento para establecer la comunicación. La utilización de esta métrica se desaconseja por el IETF desde la aparición de la MRHOF, alegando que por sí sola no representaba las bondades del enlace de la comunicación.

En el caso del RSSI tiene sentido que tenga menores pérdidas que si utilizamos el número de saltos, ya que este parámetro guarda cierta relación con el parámetro ETX, sin llegar a tener una equivalencia proporcional, pero si siendo relevante ya que la distancia que separa los nodos afectará a que se transmita correctamente o no el mensaje.

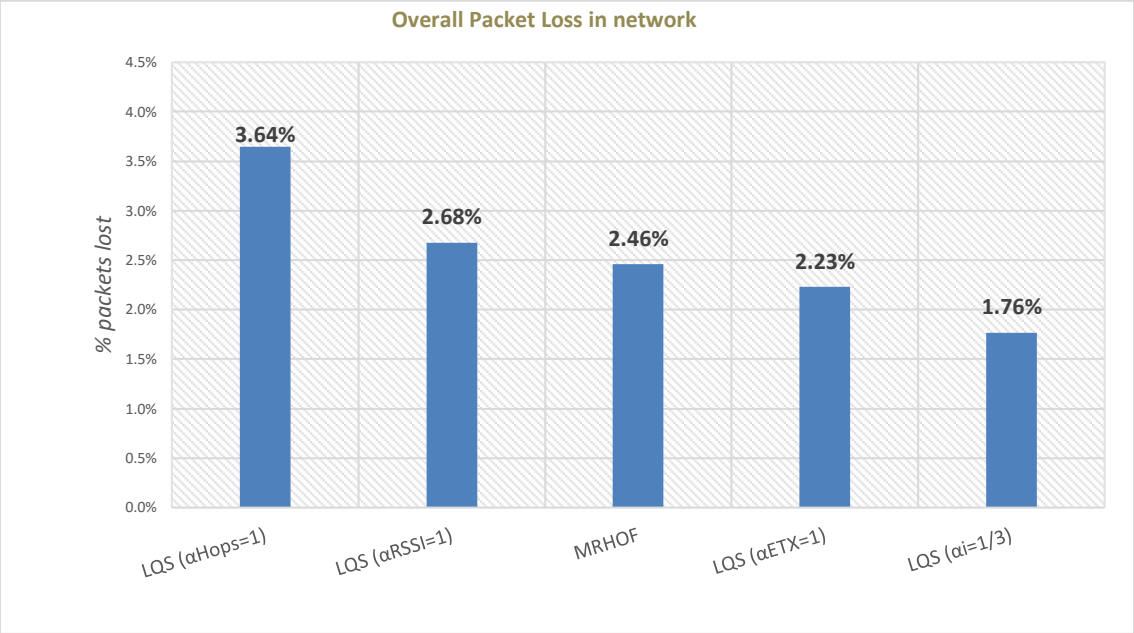


Figura 54. Pérdida de paquetes para las 5 configuraciones de O.F. simuladas.

Si comparamos ahora los resultados obtenidos para la función objetivo MRHOF junto a los obtenidos por la implementación de la función de calidad utilizando únicamente ETX como métrica, podemos ver que sus porcentajes de pérdida de paquetes son muy similares. La única diferencia entre la MRHOF y la función de calidad ($\alpha_{ETX} = 1$) está en el valor del límite máximo de métrica permitido por el enlace, configurado como 1024 para la MRHOF y con un valor de 512 en el caso de la función de calidad, que vendrá determinado por el límite de las funciones de mapeo. Estos valores representan un ETX = 4 (para 512) y de ETX = 8 (para 1024) y según la definición de la MRHOF [46] no se recomienda utilizar enlaces cuyo valor de etx sea superior a 4. Esto se entiende ya que, un ETX=1 representa un enlace en el que todos los paquetes se reciben correctamente, un ETX=2 supondrían unas pérdidas del 50% de los paquetes, obteniendo unas pérdidas del 75% de los paquetes en el caso de tener un enlace con ETX=4. Sin embargo, en la implementación de Contiki se utiliza un límite máximo para ETX, lo que explicaría que el porcentaje de pérdidas sea algo mayor utilizando MRHOF ya que se permite utilizar enlaces con peores características.

La configuración que presenta mejores resultados en cuanto al porcentaje de pérdidas es aquella en la que se utiliza la función de calidad como combinación de las tres métricas con una contribución idéntica para cada una. Este porcentaje de pérdidas es de por sí ya muy bajo en las diferentes configuraciones debido a la configuración de TSCH que se ha escogido, que debido a su funcionamiento determinista consigue paliar muchas de estas pérdidas producidas por interferencias o desvanecimientos. Aun así, se consiguen mejoras cercanas al 1% dentro de este pequeño intervalo de mejora, si lo comparamos con las otras configuraciones.

Además de evaluar el porcentaje de perdidas, se ha medido también la latencia de los mensajes desde el nodo más alejado físicamente del coordinador (nodo #7). Los resultados muestran que, de media, la configuración que utiliza una combinación de las tres métricas consigue un valor de latencia aproximadamente 3 milisegundos menor a la siguiente configuración con una latencia más baja (MRHOF). Analizando los resultados de la configuración que utiliza únicamente el número de saltos, la cual debería de obtener la menor latencia, no se corresponde con lo esperado, puesto que, aunque el camino hasta el coordinador sea más corto, los enlaces pueden tener peor comportamiento, provocando que sea necesario retransmitir más paquetes y, por lo tanto, la latencia de los mensajes sea mayor. La conclusión es que esta métrica no se aproxima lo suficiente a lo que pretendía representar, por lo que se podría cambiar por una mejor aproximación que caracterice el end-to-end delay.

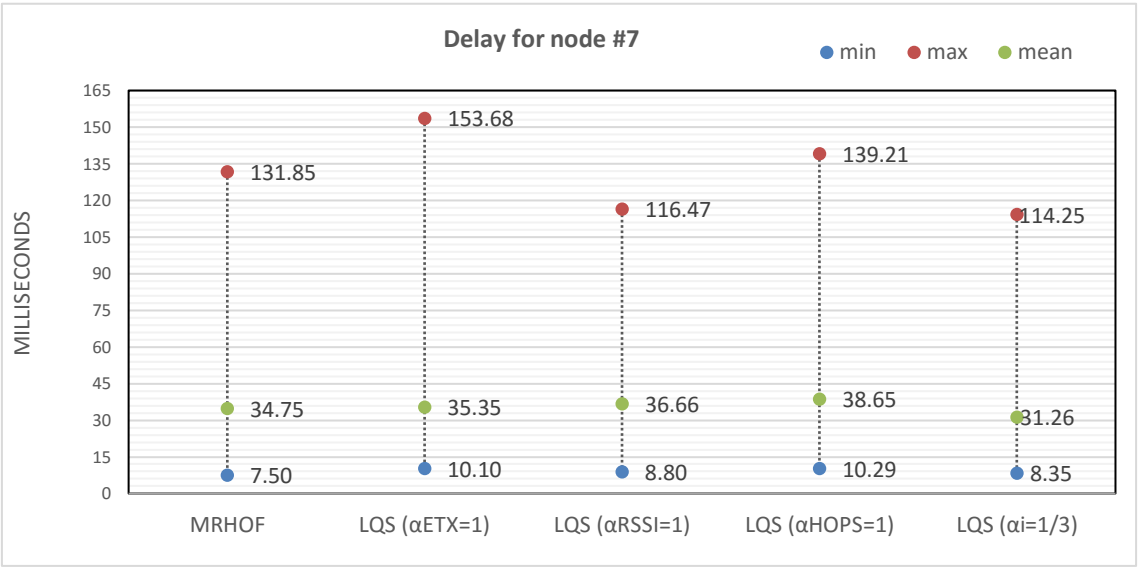


Figura 55. Caracterización del delay para las 5 configuraciones de O.F. simuladas.

Podemos decir que, mediante la utilización de la función de calidad que se ha implementado, se han conseguido mejoras significativas en cuanto a porcentaje de pérdida de paquetes, manteniendo un valor medio de latencia inferior al del resto de configuraciones. La evaluación de diferentes aspectos del enlace permite tener un mayor conocimiento del medio por el que se van a transmitir los mensajes, pudiendo actuar de manera más óptima a la hora de seleccionar la mejor alternativa de encaminamiento.

4.4 Planificación a nivel MAC-TSCH

Otro de los aspectos donde es posible optimizar el funcionamiento de las WSN, es a nivel de acceso al medio. El modo TSCH permite definir un esquema de funcionamiento determinista mediante la utilización de slots temporales que estarán sincronizados entre los dispositivos que forman parte de la red. Además, la utilización de sus características de salto en frecuencia permite paliar los problemas generados por las posibles interferencias o los desvanecimientos debidos al efecto multicamino. Sin embargo, los recursos que nos ofrece TSCH pueden ser planificados de tal forma que se obtengan mejores resultados, optimizando la utilización del espectro y de los recursos temporales. Este aspecto no es algo que venga definido por el estándar y la comunidad científica ha profundizado durante años en diferentes técnicas que permiten gestionar esta planificación de una manera eficiente.

De esta manera, queremos demostrar que es posible mejorar el funcionamiento de las WSN gestionando de manera eficiente sus recursos radio. Utilizando Orchestra como mecanismo de planificación vamos a ver cómo es posible implementar una planificación optimizada y flexible que puede ir creciendo con forme va evolucionando la red, permitiendo alojar recursos para aquellos nodos que se unan a la red. Para ello vamos a comparar el funcionamiento de Orchestra con el de diferentes modificaciones de la planificación que se plantea en Minimal 6TiSCH.

Como se describe en el Capítulo 2, la planificación de Minimal 6TiSCH utiliza un único slotframe, que puede tener una longitud configurable dependiendo del periodo de repetición que se quiera utilizar. En este slotframe estará activo únicamente un slot que estará configurado para ser utilizado de manera compartida para todos los nodos, tanto para transmitir como para recibir mensajes y llevar a cabo el proceso de sincronización de la red. Por lo tanto, tendríamos un único recurso en el que todos los nodos competirían por transmitir y que tendrán que resolver mediante los mecanismos de contienda que implementa TSCH.

Una posible modificación de la planificación de Minimal 6TiSCH es la de ampliar estos slots activos para que los dispositivos tengan más recursos disponibles sin comprometer el ciclo de trabajo que permite a los nodos ahorrar batería, ya que podríamos simplemente activarlos todos para que pudieran utilizarlos todos los nodos de la red, pero provocaría que los nodos tuviesen que permanecer siempre activos sin permitirles ahorrar batería. En la Figura 56 se muestra una posible modificación de la planificación, que añade dos recursos más, de tal forma que se configuran para que se utilicen de manera compartida por todos los nodos tanto para transmitir como para recibir, además de utilizarlo para transmitir los mensajes de control que permiten sincronizar la red.

Chann. Off. 0	TxRxS/EB	TxRxS/EB	TxRxS/EB	OFF	OFF	OFF	OFF
Chann. Off. 1	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Chann. Off. 2	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Chann. Off. 3	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SlotOffset	0	1	2	3	4	5	6

Figura 56. Modificación de planificación de Minimal 6TiSCH.

Esta solución, aunque permite que una red no muy grande y con una baja tasa de generación de paquetes pueda funcionar correctamente, la utilización de los recursos radio no es muy óptima, ya que se basa en activar un número fijo de slots para que todos los nodos de la red los compartan y necesiten entrar en contienda para acceder a ellos en el caso de que se dos dispositivos quieran acceder simultáneamente al recurso.

Por otro lado, utilizando Orchestra podemos definir una serie de normas que se ejecutarán durante el tiempo de ejecución de la red para ir habilitando los recursos necesarios en función de los nodos que se vayan conectando a la red. En nuestro caso se han configurado estas reglas para que se incluyan 3 slotframes de diferente longitud. Uno de ellos estará reservado para enviar mensajes que permitan que mantener la red sincronizada, así como advertir a nuevos nodos la configuración de la red para que puedan unirse a ella. Por lo tanto, en este primer slotframe solo se podrán enviar Beacons de señalización.

Se ha configurado un segundo slotframe que será compartido por todos los nodos de la red, y que servirá para transmitir los mensajes de control que RPL necesita para construir las topologías. Finalmente se utilizará un último slotframe donde se enviarán los mensajes de datos. Este último slotframe irá añadiendo recursos cada vez que se añada un nuevo nodo a la red, y estará basado en la dirección del nodo receptor para alojar los recursos radio necesarios. La longitud de este último slotframe de datos se ha configurado para que tenga el mismo periodo que en el caso de la modificación de Minimal 6TiSCH que hemos comentado anteriormente. En la Figura 57 se pueden ver los 3 slotframes configurados con los slots mínimos activos que tendría el coordinador al iniciar su funcionamiento. El primero, que se utilizará para transmitir las Beacons, tiene un periodo de 397 slots habilitando un único slot que se utilizará para transmitir las Beacons al resto de nodos de la red, una vez se vayan añadiendo más nodos se configurarán otros slots para recibir Beacons de otros vecinos, por lo que hasta que esto suceda los slots permanecerán inactivos y el nodo no consumirá energía esperando recibir un paquete.

El segundo slotframe es el que se utilizará para transmitir los datos, tendrá un periodo de 7 slots al igual que la configuración de Minimal 6TiSCH. Al inicio de la red solo tiene un slot activo que estará basado en la dirección del receptor (el propio nodo coordinador) por lo que cada vez que se añada un nodo a la red que sea el hijo de este nodo tendrá que configurar un slot para transmitir en la posición basada en la dirección del nodo receptor.

El último slotframe se utiliza para transmitir cualquier otro tipo de información, entre los que se encuentran los mensajes de control de RPL. Este tiene un periodo de 31 slots con un slot común habilitado para que todos los nodos puedan tanto transmitir como recibir de manera compartida.

Slotframe Handler 0	Tx/EB	OFF	OFF	…	OFF	OFF
	0	1	2	…	395	396
Slotframe Handler 1	Rx	OFF	OFF	…	OFF	OFF
	0	1	2	…	5	6
Slotframe Handler 2	TxRxS	OFF	OFF	…	OFF	OFF
	0	1	2	…	29	30

Figura 57. Planificación de 3 slotframes en Orchestra.

Una vez los nodos comiencen a añadirse a la red, estos configurarán las tres mismas planificaciones basándose en las reglas de planificación que utiliza Orchestra. Estos nodos escogerán de manera autónoma los slots que deben utilizar, basándose en la dirección de destino del mensaje, habilitando un slot para transmitir las Beacons, un slot para recibir datos de los posibles descendientes que tenga en la red y otro slot que compartirá con los nodos para transmitir los mensajes de control. Mientras estos nodos no se añadan a la red estos recursos permanecerán libres y los transceptores radio no necesitarán estar activos, a diferencia de la planificación de Minimal 6TiSCH.

Orchestra, además de permitir una mejor organización de los recursos que otras soluciones fijas como Minimal 6TiSCH, es muy flexible y escalable ya que permite modificar estas planificaciones durante el tiempo de ejecución de la red, de tal forma que añade nuevos recursos por cada nuevo nodo. En el caso de soluciones como Minimal 6TiSCH los slots permanecerían estáticos y siempre compartidos por cada vez más nodos, resultando en mayor necesidad de utilizar mecanismos de contienda y necesitando reenviar más paquetes.

Utilizando esta planificación que acabamos de describir, los nodos tendrán los siguientes slots activos:

- **Slotframe de EBs**: cada nodo tendrá slot para Tx sus propias beacons de señalización y un slot de Rx para escuchar las beacons de su origen de tiempos. Todo ello con un periodo de repetición de 397.
- **Slotframe para mensajes broadcast**: es el que utilizamos para los mensajes de control de RPL y utiliza un único slot común para todos los nodos con un periodo de repetición de 31.
- **Slotframe de datos**: cada nodo tiene dos slots planificados, uno para comunicar con su padre y otro para comunicarse con sus hijos, en el caso de que los tuviera. Por lo tanto, habrá un slot que siempre estará activo para recibir paquetes de sus hijos y un slot que se habilitará solo para transmitir a su padre.

Teniendo en cuenta esto podemos comparar el ciclo de trabajo que tendría un nodo en ambas planificaciones. Teniendo en cuenta que el periodo de generación de paquetes es de unos 6 segundos, se podrán transmitir un total de 85 slotframes de 7 slots de longitud durante ese tiempo, que para el caso de Minimal 6TiSCH supone tener 3 de los 7 slots siempre activos escuchando el canal ya que tienen que estar siempre habilitados esperando por si llega algún mensaje. Esto hace que el ciclo de trabajo para esta configuración de Minimal 6TiSCH sea del 42,86%.

Por otro lado, Orchestra utilizará 1 slotframe para las beacons, 19 slotframes para mensajes de control de RPL y 85 slotframes para transmitir datos durante ese periodo de 6 segundos. De esta

forma cada slotframe tendrá únicamente 1 slot que permanecerá siempre activo para recibir mensajes y un slot para transmitir que solo estará activo cuando sea necesario transmitir un mensaje, en este caso 1 vez cada 6 segundos. Esto hace que para Orchestra el ciclo de trabajo sea del 17,92%.

En el siguiente apartado se realizarán unas simulaciones en las que se compararán las dos planificaciones descritas en este apartado. Mostrando cómo puede afectar a las pérdidas una planificación u otra en diferentes tipos de redes.

4.4.1 Simulaciones y resultados

Para llevar a cabo las simulaciones utilizaremos Cooja, configurando un medio UDGM donde las pérdidas van aumentando con la distancia desde el nodo, reflejando así un área de cobertura e interferencia producido por cada nodo. Se ha escogido este medio de propagación debido a que no nos resulta de interés comprobar cómo afecta a la topología el cambio de las métricas de cada enlace si no como gestiona el acceso al medio de comunicación en diferentes tipos de topologías.

En las primeras simulaciones se ha configurado una topología como la que se muestra en la Figura 58, que estará formada por 8 nodos (1 coordinador y 7 clientes) en el que se transmitirán mensajes cada 6 segundos con información relevante que nos permita conocer si se han producido pérdidas durante la simulación. Los nodos se colocarán de tal forma que solo puedan alcanzar a los nodos del siguiente salto, por lo que el nodo 8 (el más alejado del coordinador) necesitará de mínimo 3 saltos para llegar hasta el coordinador.

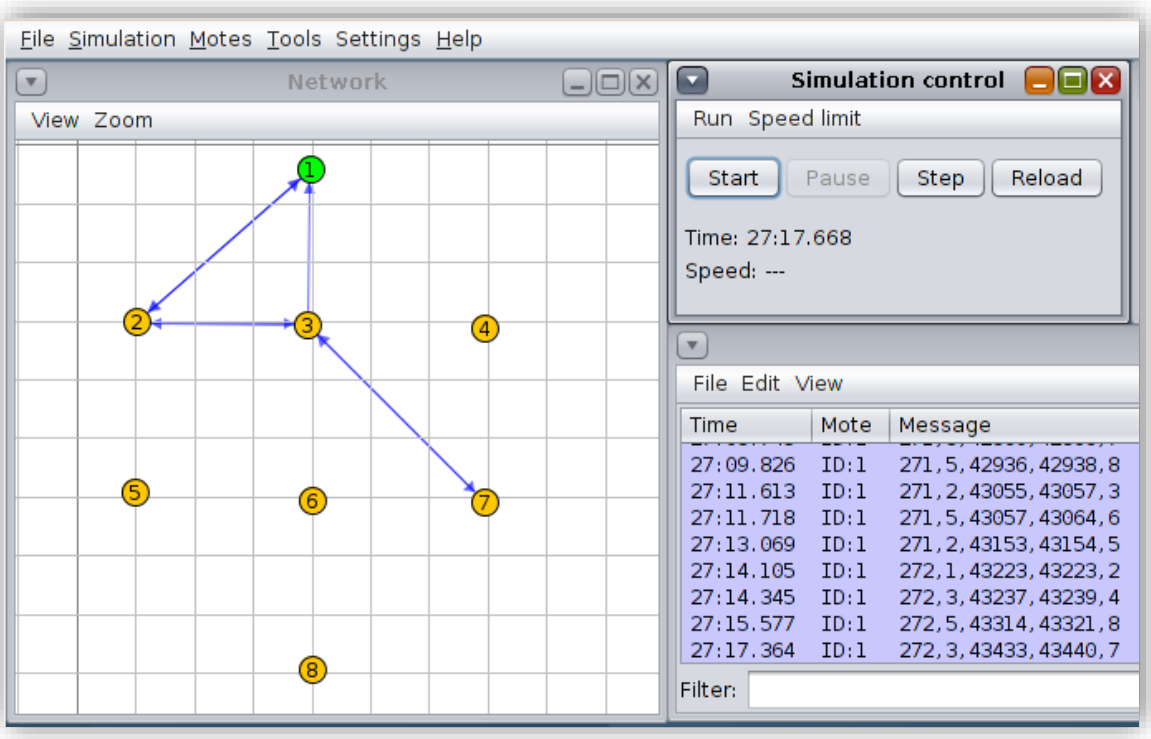


Figura 58. Topología 1 utilizada en las simulaciones con Orchestra.

Utilizando esta configuración hemos podido comprobar que no existe apenas diferencia en cuanto a pérdida de paquetes, siendo algo inferiores en Orchestra (0,04% de pérdidas) que para el caso de Minimal 6TiSCH (0,05% de pérdidas). Esta diferencia es apenas apreciable y se debe a que ambas planificaciones tienen los recursos suficientes para poder alojar los mensajes que necesita transmitir cada nodo. Recordemos que para Minimal 6TiSCH tenemos 3 slots compartidos por cada slotframe que se repetirá hasta 85 veces en un periodo de 6 segundos, por

lo que es tiempo suficiente para que los dispositivos vacíen sus colas y puedan resolver las situaciones de contienda que puedan surgir durante el intercambio de información.

Para ver si obteníamos alguna diferencia realizamos otras simulaciones en la que utilizábamos una red con más nodos colocados de manera aleatoria. Se han utilizado un total de 22 nodos (1 coordinador y 21 clientes) distribuidos de manera aleatoria, necesitando algunos de ellos de varios saltos para alcanzar al coordinador. La tasa de generación de paquetes es la misma que para la simulación anterior. En la Figura 59 se puede ver la topología simulada.

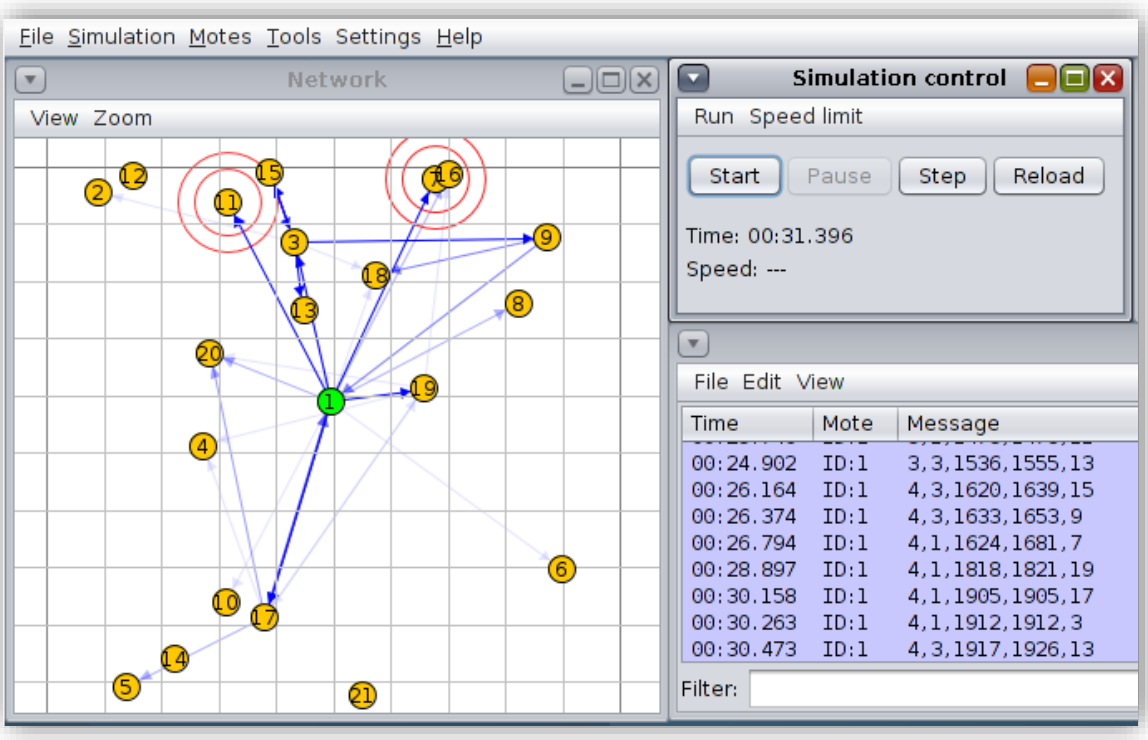


Figura 59. Topología 2 utilizada en las simulaciones con Orchestra.

Los resultados de estas simulaciones no muestran demasiadas diferencias respecto a las realizadas anteriormente. En el caso de la planificación con Minimal 6TiSCH se han obtenido pérdidas ligeramente menores a las de Orchestra con un valor del 0,46% frente al 0,82% que presenta Orchestra. A la vista de estos resultados podemos decir que la planificación modificada de Minimal 6TiSCH ofrece resultados similares en cuanto a pérdida de paquetes en este tipo de redes donde el número de nodos no es muy grande y la tasa de generación de paquetes es moderada.

Posiblemente realizando otro tipo de configuraciones de Orchestra se obtengan mejores resultados, pudiendo configurar algunos slots más dedicados al tráfico de RPL que permiten tener actualizados las características de las topologías, ya que en la planificación de Orchestra se tienen aproximadamente la mitad de recursos destinados a este tipo de tráfico. Sí que podemos decir que mediante Orchestra podemos llegar a tener una planificación que esté libre de contienda, evitando así que los nodos tengan que competir por acceder a los recursos y minimizando el número de reintentos necesarios. Además, hemos visto que con una planificación más óptima es posible reducir el ciclo de trabajo, consiguiendo que los nodos ahorren energía y se pueda extender la vida útil de la red.

Capítulo 5. Conclusiones y propuestas de trabajo futuro

5.1 Conclusiones

A lo largo de esta memoria se han descrito los diferentes mecanismos y soluciones que se han utilizado para lograr el objetivo principal descrito en los primeros capítulos. Se han realizado diferentes implementaciones que permiten construir una topología que permita extender la vida útil de la red y también que utilice información extraída de la calidad de los enlaces para escoger los enlaces que presenten una mayor calidad hasta el coordinador. Además, se han probado diferentes tipos de planificaciones de los recursos de TSCH que pueden ayudar a optimizar su utilización.

Los resultados han demostrado que, al evaluar la información sobre las baterías y como de rápido se descargan, puede permitir a RPL construir una topología que consiga balancear la carga entre los diferentes nodos que forman la red, de forma que se evite utilizar caminos que estén consumiendo rápidamente su batería. Utilizando estas topologías junto a un acceso al medio determinista como es TSCH conseguimos un que la red trabaje de manera optimizada y eficiente energéticamente, lo que permitirá que la red pueda operar de manera desatendida durante periodos de tiempo mayores. Esto ayudará a que la solución de despliegue rápido propuesta, Deploy&Forget, ayude a agilizar esas fases de despliegue y que permita que la red funcione durante más tiempo.

La evaluación de la calidad utilizando una combinación de diferentes métricas también ha dado buenos resultados ya que asegura una transmisión end-to-end fiable en escenarios industriales a través de una eficiente estimación de la calidad de los enlaces que forman el camino hasta el coordinador. Esta solución utiliza como métricas el número de retransmisiones necesarias por un enlace, el número de saltos y el RSSI de cada uno de los canales que intervienen en la comunicación, sin necesidad de realizar ningún intercambio extra de paquetes. A la vista de los resultados se han podido observar las diferentes ventajas en cuanto a fiabilidad y latencia que puede ofrecer esta solución, quedando margen de mejora permitiendo añadir otras métricas más exactas y evaluando su rendimiento en diferentes configuraciones y escenarios.

En cuanto a los mecanismos de planificación de los recursos que nos ofrece el método de acceso determinista TSCH, vemos a Orchestra como una solución prometedora que ofrece muchas posibilidades de configuración de diferentes normas que ayuden a construir la planificación de la red. Esto nos podría permitir definir diferentes tipos de tráfico con prioridades que requieran diferentes recursos radio para garantizar sus entregas. Los resultados obtenidos no muestran unas conclusiones definitivas, pero es posible realizar un estudio en mayor profundidad con diferentes topologías que permitan resaltar las bondades de este mecanismo de planificación.

5.2 Propuestas de trabajo futuro

El mecanismo de medición de calidad de los enlaces aún tiene algunos puntos de mejora, solo se han abordado algunas de las posibilidades que este puede ofrecer, realizando unas configuraciones iniciales sencillas para observar su funcionamiento dentro de un escenario determinado. La posibilidad de configuración de esta función de calidad le aporta mucha flexibilidad ya que resulta sencillo cambiar los pesos que rigen esta ecuación para dar más importancia a unas métricas frente a otras. Con esto se podría plantear la realización de un estudio de mayor alcance en el que se probaran diferentes configuraciones para observar el rendimiento que este nos podría ofrecer a la hora de seleccionar de una manera más eficiente las rutas de encaminamiento y también como medida de calidad que pueda ayudar en las etapas de despliegue.

Además, sería posible poder implementar otro tipo de métricas que ayuden a mejorar otros aspectos de la red de sensores. Este es el caso de la carga de datos que circula por un nodo, o la cola de paquetes que dispone para transmitir. Incluyendo estos parámetros podríamos ser capaces de desplegar redes que no solo escojan los mejores caminos en función de su calidad, si no que además, tengan en cuenta la carga de sus nodos vecinos para ser capaces de balancear la carga de la red, teniendo un impacto directo sobre el consumo de las baterías y por tanto en la vida útil de la red. La idea sería construir un mecanismo que fuese cambiando durante el tiempo de ejecución de la red, utilizando unas métricas más relevantes durante el proceso de despliegue y que una vez desplegadas puedan incluir de manera autónoma otras métricas como el nivel de batería o la carga de tráfico, permitiendo que la red reaccione ante los distintos eventos que puedan surgir durante el tiempo que estará funcionando la red.

Otra posible propuesta futura que se puede barajar es sobre los mecanismos de planificación de recursos del modo TSCH. Se ha visto que el mecanismo Orchestra ofrece una planificación bastante estructurada que permite gestionar los recursos de canales y slots temporales de manera eficiente en función de los vecinos y de la información que el protocolo RPL le proporciona. Sin embargo, debido a que no intercambia ningún otro tipo de información con sus vecinos estas planificaciones pueden no encajar con el modelo de red. Este podría ser el caso en que un nodo tenga bastante descendencia por debajo de él y necesite más recursos para poder transmitir todos los paquetes en comparación a un nodo que solo transmita los suyos propios. Esto podría solucionarse si a la hora de realizar la planificación se tuviera información sobre el tráfico o las colas de paquetes, dando así más recursos a aquellos nodos que lo necesiten. Ya que solo se basa en la información de RPL, se podría incluir estos valores en los Metric Container de los mensajes de control que utiliza RPL, para que los nodos tengan información actualizada sobre los parámetros relevantes para construir la topología.

Bibliografía

[1] P. Tiwari, V. P. Saxena, R. G. Mishra and D. Bhavsar. "Wireless Sensor Networks: Introduction, Advantages, Applications and Research Challenges," *HCTL Open International Journal of Technology Innovations and Research (IJTIR)*, vol 14, 2015.

[2] M. Shamsi, "Wired versus wireless trade-offs - How to choose for new installations," a publication of the international society of automation.

[3] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," *Computer Communications*, 54, 1-31, 2014.

[4] E. Mackensen, M. Lai and T. M. Wendt, "Bluetooth Low Energy (BLE) based wireless sensors," *IEEE Sensors*, 1-4, 2012.

[5] E. Mackensen, M. Lai and T. M. Wendt, "Performance analysis of a Bluetooth Low Energy sensor system," *IEEE 1st International Symposium on Wireless Systems (IDAACS-SWS)*, 2012.

[6] K. Mikhaylov, N. Plevritakis and J. Tervonen, "Performance Analysis and Comparison of Bluetooth Low Energy wit IEEE 802.15.4 and SimpliciTI," *Journal of Sensors and Actuator Networks*, 2, 589-613, 2013.

[7] M. Grover, S. K. Pardeshi, N. Singh and S. Kumar, "Bluetooth Low Energy for Industrial Automation," *2nd International Conference on Electronics and Communication Systems (ICECS)*, 2015.

[8] IEEE Computer Society, 2015. IEEE Standard for Information technology, Part 15.4; Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs).

[9] D. De Guglielmo, S. Brienza and G. Anastasi, "IEEE 802.15.4e: A survey," *Computer Communications*, 88, 1-24, 2016.

[10] Q. Wang and J. Jiang. "Comparative Examination on Architecture and Protocol of Industrial Wireless Sensor Network Standards," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2197-2219, Third quarter of 2016.

[11] IEC 62591: 2016 – Industrial networks – Wireless communication networks and communication profiles – WirelessHART™. Available online: IEC Webstore.

[12] IEC 62734: 2014 – Industrial networks – Wireless communication network and communication profiles – ISA 100.11a. Available online: IEC Webstore.

[13] IEC 62601: 2015 – Industrial networks – Wireless communication networks and communication profiles – WIA-PA. Available online: IEC Webstore.

[14] IETF RFC 8180. "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration".

[15] D. De Guglielmo, S. Brienza and G. Anastasi. "IEEE 802.15.4e: A Survey," *Computer Communications*, 88, 1-24, 2016.

[16] IETF RFC 8180, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration".

[17] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco and G. Boggia. "Traffic Aware Scheduling Algorithm for realiable low-power multi-hop IEEE 802.15.4e networks." In *Proceedings of IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, 2012.

[18] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler and T. Engel. "On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE 802.15.4e TSCH". *IEEE Sensors Journal*, 13 (10), 3655-3666, 2013.

[19] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco and M. Dohler. "Decentralized Traffic Aware Scheduling for multi-hop Low power Lossy Networks in the Internet of Things." In *Proceeding of IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2013.

[20] R. Soua, P. Minet and E. Livolant. "Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15.4e TSCH networks." *Transaction on Emerging Telecommunications Technologies*, 2015.

[21] S. Duquennoy, B. Al Nahas, O. Landsiedel and T. Watteyne. "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH." In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2015.

[22] IETF RFC 4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", last updated September 2007.

[23] IETF RFC 6282, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", last updated September 2011.

[24] IETF RFC 7400, "6LowPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", last updated November 2014.

[25] IETF RFC 6550, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", last updated March 2012.

[26] IETF RFC 5673, "Industrial Routing Requirements in Low-Power and Lossy Networks", last updated October 2009.

[27] IETF RFC 5826, "Home Automation Routing Requirements in Low-Power and Lossy Networks", last updated October 2015.

[28] IETF RFC 5867, "Building Automation Routing Requirements in Low-Power and Lossy Networks", last updated June 2010.

[29] IETF RFC 6551, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", last updated March 2012.

[30] Contiki webpage: The Open Source OS for the Internet of Things.

[31] Repositorio web de Contiki en GitHub.

[32] Lista de dispositivos hardware compatibles con Contiki.

[33] OpenWSN webpage: Open-source implementation of protocol stacks based on Internet of Things standars.

[34] Zolertia RE-Mote Resources. Incluye diferentes descripciones del producto, así como pequeñas guías de utilización de los dispositivos.

[35] M.Stehlik, “Comparison of Simulators for Wireless Sensor Networks”. *PhD thesis*, Masaryk University, 2011.

[36] H. N. Pham, D. Pediaditakis and A. Boulis. “From Simulation to Real Deployments in WSN and Back,” *World of Wireless, Mobile and Multimedia Networks*, 1-6, 2007.

[37] R. Szewczyk, J. Polastre, A. M. Mainwaring and D. E. Culler. “Lessons from a sensor network expedition”. In *H. Karl, A. Willig and A. Wolisz editors, EWSN*, volume 2920 of Lecture Notes in Computer Science, pages 307-322. Springer, 2004.

[38] S. Santonja, V. Sempere: Patent P2015.300.39. Method of rapid deployment of nodes in a network and node to implement said method. Spain, granted on August 2016.

[39] D. Todolí, J. Silvestre-Blanes, S. Santonja-Climent, V. Sempere-Paya, J. Vera-Pérez. “Deploy&Forget Wireless Sensor Networks for itinerant applications”. Computer Standards & Interfaces, granted on August 2017.

[40] A. Guinard, A. McGibney, D. Pesch. “A wireless sensor network design tool to support building energy management”. *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, 2009, Berkeley, California, November 03, 2009, pp. 25-30.

[41] Plug & Play & Forget ® technology by IK4-TEKNIKER. Link: Plug&Play&Forget.

[42] T. C. Huang, H. R. Lai, C. H. Ku. “A deployment procedure for Wireless Sensor Networks”. *The 2nd Workshop on Wireless, Ad Hoc, and Sensor Networks*, 2006.

[43] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli. “The Hitchhiker’s Guide to Successful Wireless Sensor Network Deployments”. *Proceedings of the 6th ACM conference on Embedded network systems (2008)*.

[44] J. Lloret. “Introduction to Practical Deployments on Wireless Sensor Networks”. In *International Journal on Advances in Networks and Services*, vol 3 no 1 & 2, 2010.

[45] A. Dunkels, J. Eriksson, N. Finne, N. Tsiftes. “Powertrace: Network-level power profiling for low-power wireless networks”, 2011.

[46] IETF RFC 6719, “The Minimum Rank with Hysteresis Objective Function”, last updated September 2012.

[47] J. Vera-Pérez, D. Todolí-Ferrandis, J. Silvestre-Blanes, V. Sempere-Paya and S. Santonja-Climent. “Path Quality Estimator for Wireless Sensor Networks fast deployment tool”, *25th Telecommunications forum TELFOR*, November 2017. Pendiente de aceptación.

[48] IETF RFC 6552, “Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)”, last updated March 2012.

Lista de figuras

Lista de tablas