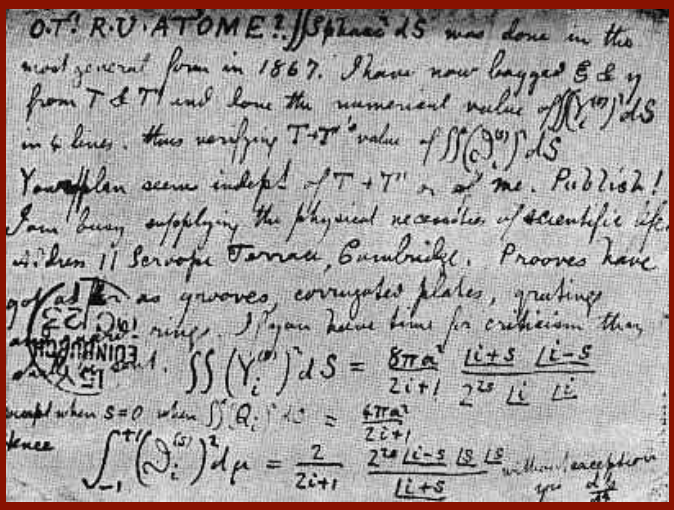
Tesis Doctoral
Ingeniería de Telecomunicación

Contribuciones al ahorro de energía en dispositivos IoT IEEE 802.15.4 con RPL y potencia de transmisión ajustable



Autor: Juan A. Ternero Muñiz
Director: Rafael Estepa Alonso

Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Tesis Doctoral
Ingeniería de Telecomunicación

Contribuciones al ahorro de energía en dispositivos IoT IEEE 802.15.4 con RPL y potencia de transmisión ajustable

Autor:

Juan A. Ternero Muñiz

Director:

Rafael Estepa Alonso

Profesor Titular

Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

2024

Tesis Doctoral: Contribuciones al ahorro de energía en dispositivos IoT IEEE 802.15.4 con RPL y potencia de transmisión ajustable

Autor: Juan A. Ternero Muñiz
Director: Rafael Estepa Alonso

El tribunal nombrado para juzgar la Tesis arriba indicada, compuesto por los siguientes doctores:

Presidente:

Vocales:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

En primer lugar, esta tesis no hubiera sido posible sin mi director de tesis, su guía, optimismo y empuje han sabido sacar de mí las fuerzas necesarias para conseguir un logro que me parecía inalcanzable. Muchas gracias.

Quiero también agradecer su apoyo y ayuda a mis compañeros, y sin embargo amigos, del Depto. de Ingeniería Telemática.

Por último, y no menos importante, doy las gracias a mi familia, que me ha ayudado y soportado durante el largo camino de la elaboración de esta tesis.

Juan Antonio Ternero Muñiz
Sevilla, 2024

Resumen

En esta tesis se aborda el problema de disminuir el consumo total de energía en una red inalámbrica de dispositivos IoT sin perjuicio de la calidad del servicio. Se asume que los dispositivos IoT utilizan un enlace radio IEEE 802.15.4 que ofrece varios niveles de potencia de transmisión, y que usan el protocolo de encaminamiento RPL para formar la red.

La solución propuesta resuelve simultáneamente el problema de encaminamiento en la red y la selección óptima de la potencia de transmisión en cada nodo, haciendo que los mensajes lleguen a su destino minimizando el gasto energético posible y manteniendo la calidad del servicio de red. Para ello, se emplea en cada nodo un enfoque de capa cruzada que utiliza información intercambiada por los protocolos de nivel de red (encaminamiento RPL), de acceso al medio y de nivel físico. Cabe señalar la interdependencia entre la potencia de transmisión y el alcance y la calidad en los enlaces radio, y por lo tanto, entre el camino óptimo en la red y la potencia de transmisión en cada nodo. Por ello, es conveniente la selección conjunta de ambos parámetros de operación.

Dada la complejidad del problema, y las limitaciones de recursos existentes en los nodos IoT, se ha propuesto un algoritmo de bajo consumo de recursos computacionales implementado sobre el protocolo de encaminamiento RPL para la selección óptima de ambos parámetros. A fin de reducir el consumo energético ligado al plano de control de RPL, esta tesis también propone un nuevo mecanismo alternativo de sondeo de alcanzabilidad de los vecinos, que permite mantener frescas las estadísticas de comunicaciones de la capa de enlace, pero restringiendo el número de mensajes de control.

La solución propuesta se ha implementado y validado en dispositivos IoT con el sistema operativo Contiki (motas Z1), con hasta 5 niveles de potencia de transmisión, usando el simulador Cooja y realizando simulaciones con 15 dispositivos distribuidos en áreas de diferentes tamaños: desde $10m \times 10m$ ($7m^2$/dispositivo), hasta $100m \times 100m$ ($667m^2$/dispositivo). Los resultados muestran que se consigue un ahorro significativo comparándolo con la transmisión a un nivel de potencia máximo (RPL estándar) o con la transmisión a dos niveles de potencia (RPL binario), logrando una reducción de hasta un 20 % en la transmisión y un 34 % en la recepción.

Abstract

This thesis addresses the challenge of reducing the total energy consumption in a wireless network of IoT devices without compromising quality of service. It is assumed that IoT devices utilize an IEEE 802.15.4 radio link offering various transmission power levels and employ the RPL routing protocol for network formation.

The proposed solution concurrently addresses both the routing problem in the network and the optimal selection of transmission power at each node. This ensures that messages reach their destination by minimizing energy expenditure and maintaining network service quality. To achieve this, a cross-layer approach is employed at each node, utilizing information exchanged by network layer protocols (RPL routing), medium access layer, and physical layer. The interdependence between transmission power, range, and link quality in radio links, and consequently, between the optimal path in the network and the transmission power at each node, is noteworthy. Thus, the joint selection of both operational parameters is advisable.

Given the complexity of the problem and the resource limitations in IoT nodes, an algorithm with low computational resource consumption is proposed and implemented on the RPL routing protocol for the optimal selection of both parameters. To mitigate the energy consumption associated with the RPL control plane, this thesis introduces a novel alternative neighbor reachability probing mechanism. This mechanism maintains fresh link layer communication statistics while restricting the number of control messages.

The proposed solution has been implemented and validated on IoT devices running the Contiki operating system (Z1 motes), featuring up to 5 transmission power levels. Simulations were conducted using the Cooja simulator with 15 devices distributed in areas of various sizes: from $10m \times 10m$ ($7m^2$/device) to $100m \times 100m$ ($667m^2$/device). The results demonstrate significant energy savings compared to transmission at maximum power level (standard RPL) or transmission at two power levels (binary RPL), achieving a reduction of up to 20 % in transmission and 34 % in reception.

Índice

1 Introducción y objetivos

El término "Internet de las Cosas" (IoT o Internet of Things) se refiere a una red de dispositivos físicos, vehículos, aparatos electrónicos y otros objetos físicos que están equipados con sensores, software y conectividad de red, lo que les permite recopilar y compartir datos [1]. Actualmente está en auge su aplicación en distintos campos, como por ejemplo, en las ciudades inteligentes (smart cities) con sistemas de control del alumbrado, información de aparcamiento, medición de contaminación, control de tráfico, detección del llenado de contenedores de elementos reciclables, etc [2]. También se aplica en entornos no urbanos, como en la agricultura de precisión (plantaciones, campos de golf), detección de catástrofes (incendios forestales, inundaciones cerca de ríos), calidad del entorno (gases en granjas, radiación UV en playas, potabilidad de acuíferos), etc [3].

Las redes de sensores inalámbricos (WSN o Wireless Sensor Networks) forman una parte importante del IoT, y están formadas por pequeños dispositivos electrónicos autónomos, también llamados motas o nodos, que se comunican entre ellos mediante redes inalámbricas, para transmitir los datos recopilados por sensores. Estos nodos suelen ser bastante simples, con restricciones de memoria y procesamiento, buscando el bajo coste para que sea rentable su uso masivo, y alimentados por batería para poder ubicarlos fácilmente en exteriores allí donde el sensor necesite hacer la medición. Debido a ello, los enlaces radio utilizados en estos dispositivos son de bajo consumo y suelen ser aplicados en entornos con ruido. Estos nodos tan simples no se conectan directamente a Internet, sino que la conexión se realiza a través de un nodo central llamado nodo raíz, y el resto de los nodos se comunican con el nodo raíz para enviar la información al exterior. En despliegues extensos, la configuración de la red no es en estrella [4], ya que la radio no tiene el alcance suficiente para que todos los nodos puedan comunicarse con el nodo raíz directamente, sino que la configuración es ad-hoc, de tal forma que los mensajes que envían los sensores hacia el nodo raíz tienen que atravesar nodos intermedios. Para determinar el camino que deben seguir los mensajes hay que utilizar en el nivel de red un protocolo de encaminamiento que tenga en cuenta las características de comunicaciones de estas redes, entre ellas la potencia y las pérdidas.

El protocolo de encaminamiento RPL [5] está normalizado por el IETF y ha sido especialmente diseñado para redes de bajo consumo y mucho ruido ("Low Power and Lossy Networks") (LLN), como las que se usan en algunos escenarios del IoT. Mediante este

protocolo, cada nodo elige un nodo padre y se forma un árbol que permite la comunicación con el nodo raíz. Se pueden usar distintas funciones objetivo para ello, por ejemplo para disminuir el número de saltos o para minimizar el número de retransmisiones previstas.

Entre los distintos propósitos de la función objetivo, se puede destacar el de disminuir el consumo de energía total, ya que si se agota la batería del nodo hay que reemplazarla, o incluso reemplazar el propio nodo [6]. Este reemplazo puede tener un coste similar al despliegue, debido al desplazamiento y a la manipulación física por parte del personal, siendo este coste uno de los parámetros que más influye en el uso de redes de sensores [7]. Teniendo en cuenta lo anterior, cualquier mecanismo que permita el ahorro de energía en las redes de sensores inalámbricos alimentados por batería, implica una importante disminución del coste asociado a su uso.

Para calcular el consumo de energía hay que tener en cuenta, en cada nodo, la potencia consumida debido a las (re)transmisiones de los paquetes generados o reenviados por el nodo (además de los recibidos). Estos factores pertenecen a los 3 niveles más bajos del modelo OSI (red, enlace y físico), y por lo tanto un enfoque de capa cruzada (cross-layer), que sea capaz de tener en cuenta simultáneamente la información obtenida de esos tres niveles, permite obtener, previsiblemente, mejores resultados que si se abordan por separado.

En el nivel de enlace y físico, las comunicaciones inalámbricas entre los sensores IoT suelen estar basadas en el estándar IEEE 802.15.4 [8] (aunque existen otros protocolos con menor prevalencia), que es la norma del IEEE que define la capa física y la subcapa de control de acceso al medio (Medium Acces Control o MAC layer) para las redes de área personal inalámbrica de baja tasa (LR-WPAN o low-rate wireless personal area network), y que permite la transmisión a distintas potencias. El hecho de poder transmitir a distintas potencias implica que se pueda transmitir a un nivel inferior al máximo y consumir menos energía, lo que podría afectar al alcance de transmisión y, por consiguiente, a la calidad de servicio.

1.1 Objetivos

En esta tesis se van a estudiar y proponer algoritmos y mecanismos aplicados a RPL, usando distintos niveles de potencia de transmisión, teniendo en cuenta también información de la capa física y subcapa MAC de la norma IEEE 802.15.4, con el objetivo de reducir la energía total consumida y mantener la calidad de servicio en las comunicaciones de redes de sensores inalámbricos.

Los resultados de este trabajo serán aplicables a redes de sensores inalámbricos que puedan transmitir a distintos niveles de potencia, que en la comunicación radio se basen en la norma IEEE 802.15.4 y que usen el protocolo de encaminamiento RPL.

Para lograr este objetivo, se fijan una serie de objetivos parciales:

- Modificar el mecanismo de transmisión para que el protocolo RPL sea quien gobierne dinámicamente el nivel de potencia de transmisión del nodo, dentro del rango de potencias posibles.
- Modificar el algoritmo de sondeo usado por RPL para que el uso de múltiples niveles de potencia no implique un incremento exponencial de los mensajes de señalización.

- Extender la selección del nodo padre del protocolo RPL teniendo en cuenta los distintos niveles de potencia de transmisión.

1.2 Metodología

Para alcanzar los objetivos propuestos se han seguido los siguientes pasos:

- Análisis de las comunicaciones. Se estudian las tres capas inferiores del modelo OSI y los protocolos usados en las redes de sensores inalámbricas.
- Definición de una adaptación del protocolo RPL para conseguir la reducción de la energía total en las motas, haciendo uso de múltiples niveles de potencia.
- Definición de un algoritmo alternativo de sondeo en RPL, a fin de reducir el consumo de energía del tráfico de control, cuando se emplean múltiples niveles de potencia.
- Análisis y modificación del sistema operativo Contiki-NG, ampliamente usado en motas comerciales, para implementar los algoritmos propuestos.
- Validación del rendimiento de los algoritmos propuestos mediante simulación con Cooja en distintos escenarios IoT. Cooja es un simulador escrito en Java, que permite simular un conjunto de motas con el sistema operativo Contiki-NG.

2 Conocimientos previos y estado del arte

En este capítulo se van a exponer las características más relevantes del IoT y de RPL, y también las trabajos relacionados con los mecanismos propuestos en esta tesis.

2.1 IoT

El Internet de las Cosas (IoT) es un término genérico que engloba la interconexión de múltiples dispositivos y objetos cotidianos a través de Internet. El IoT contiene múltiples sistemas interconectados donde los nodos que generan información suelen ser equipos con poca capacidad de proceso y que están conectados a Internet bien directamente o bien a través de una pasarela que hace de agregador de varios de esos equipos [9]. Las peculiaridades de cada instalación y de los dispositivos empleados (con recursos muy limitados) hacen que no exista un conjunto de protocolos únicos en el campo del IoT. En esta tesis consideraremos principalmente la pila de protocolos IoT propuesta por el IEFT, mostrada en la Figura 2.1 [10].

En los siguientes apartados se introducirán los protocolos y normas del modelo del IETF, especificados por cada capa del modelo OSI.

2.1.1 Capa de aplicación

En el ámbito de la Internet de las Cosas (IoT), la capa de aplicación posibilita servicios en aplicaciones distribuidas mediante protocolos específicos, la gestión de datos adaptados al IoT, y la interoperabilidad en entornos heterogéneos, permitiendo la comunicación efectiva entre dispositivos diversos en el ecosistema del IoT.

En uso del protocolo CoAP (Constrained Application Protocol) [11] en la capa de aplicación posibilita de interacción con recursos distribuidos identificados por URI (Identificador de Recursos Uniforme [12]), y proporciona un mecanismo de observación y notificación. Estas características lo convierten en una opción adecuada para aplicaciones distribuidas en entornos del IoT. El protocolo CoAP es adecuado para redes de sensores inalámbricos,

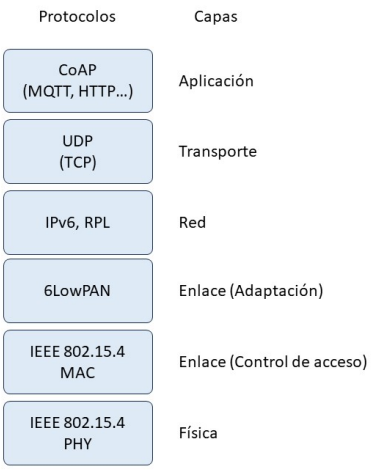


Figura 2.1 Protocolos usados en el IoT y capas.

ya que está diseñado específicamente para dispositivos y redes con recursos limitados, como dispositivos de baja potencia y ancho de banda reducido. CoAP se puede considerar una versión simplificada de HTTP, pero a diferencia de HTTP, CoAP envía sus mensajes comprimidos usando el servicio ofrecido por UDP, eliminando toda la sobrecarga de TCP (que ofrece un servicio confiable), lo que reduce los requisitos de ancho de banda, proporciona más simplicidad y lo hace más adecuado para aplicaciones del IoT [13].

También se pueden usar otros protocolos como MQTT_SN (Message Queuing Telemetry Transport for Sensor Networks) [13], AMQP (Advanced Message Queuing Protocol) [14] o incluso HTTP (Hypertext Transfer Protocol) [15], pero en general estos protocolos necesitan operar sobre TCP en vez de UDP, lo que hace que la pila de protocolos consuma más recursos computacionales, algo que se desea evitar en las redes de sensores.

2.1.2 Capa de transporte

La capa de transporte posibilita el envío y recepción de mensajes entre las aplicaciones que se ejecutan en los dispositivos finales, y en el IoT es importante tener en cuenta la interoperabilidad y la gestión de recursos limitados. En la capa de transporte de las WSN, se suele escoger el protocolo UDP [16] frente a TCP [17] debido a que UDP tiene menor sobrecarga (ya que TCP tiene que garantizar la entrega ordenada y sin pérdidas de los datos) y menor latencia (TCP establece una conexión antes de la transferencia de datos). Al tener menos sobrecarga y ser más simple en su implementación, UDP puede consumir menos energía que TCP, que es muy importante en las WSN, donde la conservación de la energía es esencial para prolongar la vida útil de los nodos alimentados por batería. El posible inconveniente de que UDP no garantiza la entrega de paquetes ni el orden de llegada puede ser aceptable en los casos donde la pérdida ocasional de datos no es crítica y la retransmisión puede ser innecesaria. Por ejemplo, en aplicaciones de monitorización

ambiental, donde la pérdida de algunos datos no es crítica, UDP puede ser una opción adecuada, y además protocolos de la capa de aplicación como CoAP ya están preparados para ello.

2.1.3 Capa de red

La capa de red consigue la comunicación de los dispositivos, atravesando los sistemas intermedios, y abarca, entre otros aspectos, el encaminamiento y el direccionamiento, teniendo en cuenta también la diversidad de tecnologías que se pueden encontrar en el IoT. En la capa de red, el uso del protocolo IPv6 [18] en las WSN aporta ventajas interesantes frente a IPv4 [19] como la autoconfiguración de direcciones (que simplifica la configuración de direcciones y facilita la implementación y la gestión), la asignación de direcciones únicas a cada dispositivo gracias al mayor espacio de direccionamiento (permitiendo incluso la comunicación directa con otros nodos conectados a Internet sin necesidad de realizar traducción de direcciones), la cabecera simplificada (se han eliminado varios campos que eran parte de IPv4) o ser el soporte para el protocolo de encaminamiento RPL (se explica a continuación). El posible inconveniente del incremento del tamaño de las cabeceras se soslaya con el uso de 6LowPAN, tal como se explica en el apartado 2.1.4.

El protocolo de encaminamiento RPL está especificado por el IETF en una RFC [5] cuyo título es: "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks". Tal como refleja su título, está diseñado para ser usado con IPv6 y es especialmente adecuado para las WSN, donde son usuales las pérdidas de paquetes y una alta variabilidad en la calidad de los enlaces. También puede adaptarse a las condiciones cambiantes de la red y escalar eficientemente, lo que es esencial en entornos dinámicos como las WSN. RPL se explica con más detalle en el apartado 2.2.

2.1.4 Capa de adaptación al enlace de datos

Uno de los problemas que presenta el uso de IPv6 en las WSN es que el nivel de enlace usado en esta tesis sólo soporta una unidad máxima de transmisión (MTU) de 127 bytes, mientras que el protocolo IPv6 requiere un tamaño de datagrama mínimo de 1280 bytes, aproximadamente 10 veces superior a la MTU. Para resolver este problema el IETF especificó una capa de adaptación llamada 6LoWPAN [20]. En particular, la capa de adaptación 6LoWPAN define los mecanismos para comprimir las cabeceras de IPv6 y fragmentar y reensamblar los paquetes IPv6, para que el datagrama pueda ser transportado dentro de las tramas de nivel de enlace, de menor MTU.

2.1.5 Capa de enlace y física

En la capa de enlace y física, las comunicaciones inalámbricas entre los sensores IoT suelen estar basadas en el estándar IEEE 802.15.4 [8]. Esta norma es apropiada en las WSN debido a su enfoque en dos aspectos básicos: bajo consumo de energía y bajo costo. Está diseñada para permitir que los dispositivos funcionen eficientemente con fuentes de energía limitadas, como baterías, y al mismo tiempo ofrece una implementación económica en dispositivos de bajo costo. Tal como dice la propia norma: "La norma proporciona conectividad inalámbrica de ultra bajo costo, ultra baja complejidad, consumo de energía ultra bajo y baja velocidad de datos entre dispositivos económicos, enfocándose

especialmente en los requisitos de comunicación de lo que ahora comúnmente se conoce como el Internet de las Cosas (IoT)".

En la capa física, la norma define varios modos de operación, pero el más utilizado es el que opera en la banda de 2450 MHz basado en DSSS (direct sequence spread spectrum), con una modulación O-QPSK (offset quadrature phase-shift keying), que permite una tasa binaria de 250 kbps. Este modo de operación es el más habitual porque opera en un rango de frecuencia libre en cualquier parte del mundo. El inconveniente de este modo es la gran cantidad de dispositivos que operan en este rango, provocando más interferencias que en las otras bandas [21].

Los métodos de acceso al medio de la subcapa MAC se pueden resumir en:

- CSMA/CA (carrier sense multiple access with collision avoidance). En este método cada estación escucha el canal durante un tiempo y se requiere que éste esté libre antes de iniciar la transmisión. Si el canal está libre, la estación espera un tiempo (Inter-frame gap) y después comienza la transmisión. Tras completar el envío de la trama, se espera la recepción de un asentimiento. Este método puede usarse con o sin tramas baliza, y con o sin acceso prioritario al canal. En esta tesis se utilizará este método (sin tramas baliza y sin acceso prioritario).
- TSCH (time slotted channel hopping). Es un método de acceso al medio con canal ranurado en tiempo y saltos de frecuencia. Este método permite ser usado con ranuras compartidas o no.
- ALOHA con acceso prioritario al canal, para la monitorización de infraestructuras críticas en bajo consumo energético.

De estos métodos, CSMA/CA sin tramas baliza y sin acceso prioritario es el más usado, por su simplicidad. La transmisión de tramas baliza permite disminuir algo el consumo de energía, pero para ello hay mecanismos más eficientes, como TSCH (explicado en el apartado 2.3.1).

También hay otros estándares usados por la industria [22], basados en IEEE 802.15.4, como son Zigbee (Connectivity Standards Alliance) [23] o Wi-SUN (Smart Utility Networks Alliance) [24], que cubren la capa de enlace y física (además de otras capas).

2.2 RPL

Como ya se ha introducido anteriormente, RPL [5] es un protocolo de encaminamiento sobre IPv6 basado en vectores de distancia que fue diseñado por la comunidad IETF en 2012 para cumplir con los requisitos de encaminamiento de un amplio número de aplicaciones en LLN ("Low Power and Lossy Networks"). Es compatible con diversas tecnologías de nivel de enlace, pero se usa típicamente con las normalizadas en la IEEE 802.15.4 [8], que es común en los dispositivos alimentados por baterías en los que no se puede cambiar la batería frecuentemente.

RPL particularmente está optimizado para aplicaciones de recolección de datos (es decir, tráfico MP2P "Multi-Point-to-Point"), proporcionando también un soporte razonable para el tráfico P2MP ("Point-to-Multi-Point"), permitiendo entre ambos un soporte indirecto de

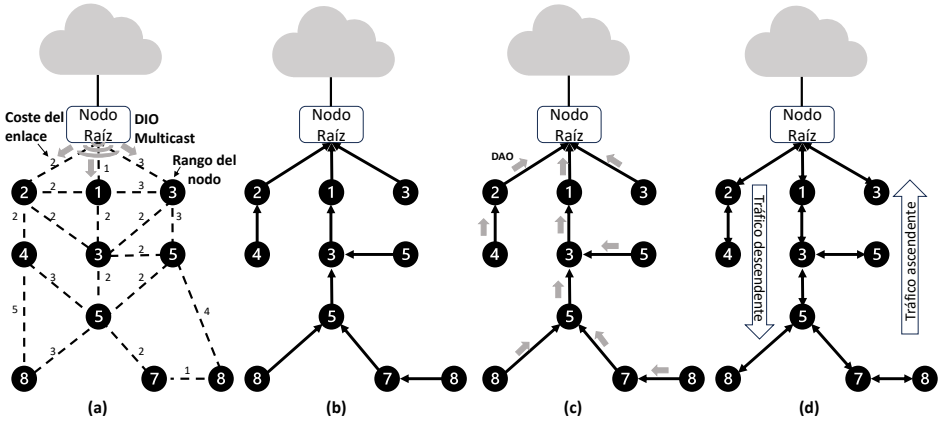


Figura 2.2 Construcción de rutas ascendentes y descendentes.

tráfico P2P ("Point-to-Point"). En esta tesis se estudiará el tráfico MP2P, suponiendo que los dispositivos IoT envían datos periódicamente al nodo que se conecta a Internet, usando la tecnología de nivel de enlace de IEEE 802.15.4 y la pila de protocolos IPv6 de IETF. En este apartado se hará sólo una introducción de RPL ya que en capítulos posteriores se desarrollarán algunos de sus aspectos más en detalle.

2.2.1 Topología y funcionamiento de RPL

RPL organiza su red física en forma de grafos acíclicos dirigidos (Directed Acyclic Graphs, DAG), donde cada DAG está orientado hacia un único destino y en términos de RPL es llamado DAG orientado al destino (Destination-Oriented DAG, DODAG). El DODAG representa el destino final del tráfico dentro del dominio de la red y conecta esta topología con otros dominios IPv6 como Internet. En RPL este nodo frontera se denomina nodo raíz o raíz DODAG.

RPL usa el término rutas ascendentes para referirse a las rutas que llevan el tráfico desde los nodos que no son el raíz hasta el nodo raíz (es decir, MP2P), por otro lado, las rutas que llevan el tráfico desde el nodo raíz hasta los demás nodos (es decir, P2MP) son llamadas rutas descendentes. Para construir las rutas ascendentes, cada nodo dentro de una red debe seleccionar a uno de sus vecinos como padre preferido (siguiente salto) hacia el nodo raíz. De manera similar, cada nodo que esté dispuesto a participar en el encaminamiento descendente debe anunciarse a uno de sus padres, preferiblemente al padre preferido. Los detalles de la construcción de las rutas ascendentes y descendentes se verán en los siguientes apartados y en la Figura 2.2 se ilustra el funcionamiento de ambas.

RPL usa el término instancia para referirse a los múltiples DODAG que comparten las mismas políticas y mecanismos de encaminamiento. Múltiples instancias de RPL podrían coexistir concurrentemente en una topología física específica y un nodo podría unirse a

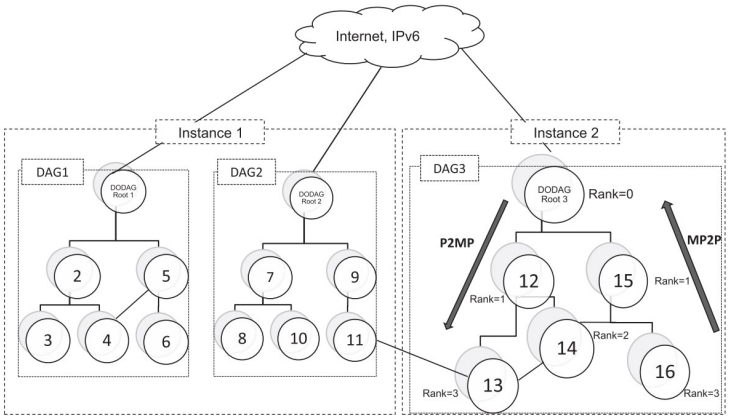


Figura 2.3 Red RPL con tres DODAG en dos instancias [25].

más de una instancia al mismo tiempo. Sin embargo, dentro de cada instancia, un nodo solo puede asociarse a un único DODAG con un único nodo raíz, aunque existan diferentes DOGAG con distintos nodos raíz cada uno. En la Figura 2.3 se puede observar un caso particular de un nodo unido a dos instancias.

Para compartir la información que permite construir la topología de red y las rutas de encaminamiento, RPL introduce cuatro mensajes ICMPv6 de control (excluyendo los mensajes de seguridad) que se detallan a continuación:

- DODAG Information Object (DIO): los mensajes DIO son usados para llevar la información relevante y la configuración de los parámetros que permiten a un nodo descubrir la instancia RPL, unirse a un DODAG especifico, seleccionar un conjunto de padres candidatos, y mantener el DODAG.
- Destination Advertisement Object (DAO): este mensaje de control permite a un nodo propagar por las rutas ascendentes su información de destino a lo largo del DODAG hasta el nodo raíz, para que las rutas descendentes, desde el nodo raíz hasta los nodos asociados, puedan ser construidas.
- DODAG Information Solicitation (DIS): este mensaje es usado por un nodo RPL para solicitar un mensaje DIO a los nodos vecinos con el objetivo de unirse al DODAG.
- Destination Advertisement Object Acknowledgement (DAO-ACK): el DAO-ACK se utiliza para asentir la recepción de los mensajes DAO a los nodos que los envíen.

El formato de los mensajes de control se puede apreciar en la Figura 2.4. En la Figura 2.5 se muestra el formato de los mensajes DIO, y en la Figura 2.6 el de los mensajes DAO.

Rutas ascendentes (Construyendo la topología DODAG)

El proceso de construcción del DODAG y de las rutas ascendentes está controlado por los mensajes DIO. Además de otra información de encaminamiento, los mensajes DIO

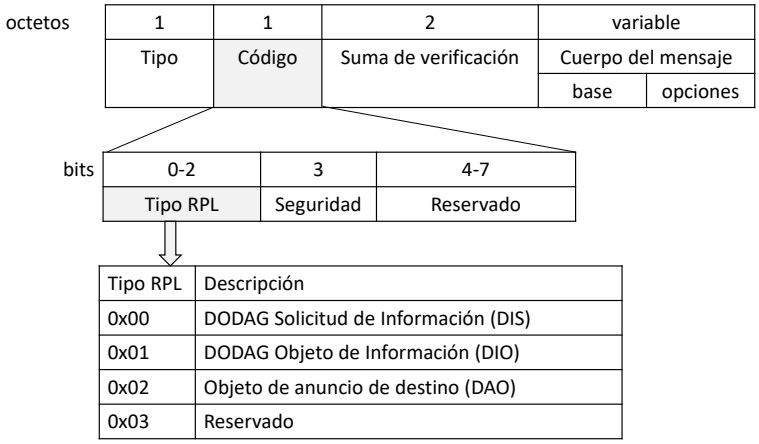


Figura 2.4 Mensaje de control RPL.

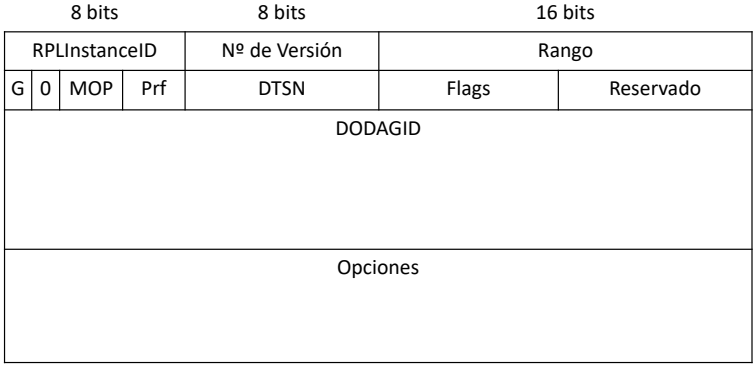


Figura 2.5 Formato de mensaje DIO.

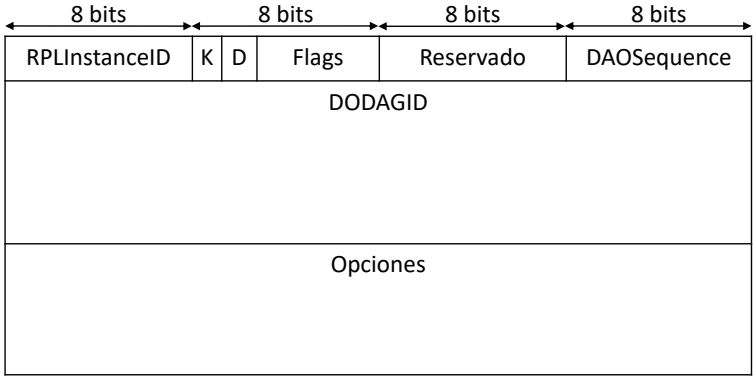


Figura 2.6 Formato de mensaje DAO.

contienen el rango (rank), el cuál es la posición relativa de un nodo RPL con respecto al nodo raíz del DODAG, y una política de encaminamiento llamada la Función Objetivo (Objective Function, OF) que especifica cómo un nodo RPL calcula el rango y elige a su padre preferido (los detalles del rango y la OF serán explicados en el apartado 2.2.2).

Expresamente, la construcción del DODAG es iniciada por el nodo raíz del DODAG cuando empieza a transmitir mensajes DIO multicast a sus nodos vecinos anunciando su rango y la función objetivo que debe ser usada. Esto se representa en la Figura 2.2a. Cuando un nodo RPL recibe un mensaje DIO realiza las siguientes acciones:

- Añade la dirección origen al conjunto de padres candidatos.
- Calcula su propio rango.
- Selecciona al padre preferido de su conjunto de padres candidatos.
- Actualiza el mensaje DIO recibido con su propio rango y lo transmite (multicast) a sus nodos vecinos para que conozcan su rango.

Un nodo podría también descartar un mensaje DIO recibido basándose en algún criterio definido en la especificación de RPL. Este proceso no termina hasta que todos los nodos tienen configurados sus rutas ascendentes hacia el nodo raíz como muestra la Figura 2.2b.

Rutas descendentes

Si se quiere facilitar los modelos de comunicación P2MP y P2P, también se deben establecer y mantener las rutas descendentes. RPL usa los mensajes DAO para este propósito. Un nodo RPL dispuesto a anunciarse a sí mismo como un destino alcanzable desde el punto de vista del nodo raíz, envía un mensaje DAO unicast a su padre preferido para anunciar su propio prefijo de destino. El procesado del mensaje DAO recibido por el padre depende del modo de operación que se anuncia en los mensajes DIO. Para tal fin, RPL ha especificado dos modos para crear y mantener rutas descendentes, llamados, storing (encaminamiento por tablas salto a salto) y non-storing (encaminamiento en el origen). En el modo storing, cuando un padre recibe un mensaje DAO de unos de sus hijos:

- Almacena el prefijo del destino anunciado en su tabla de encaminamiento, junto con la dirección del nodo que le envió el mensaje DAO como siguiente salto para alcanzar ese destino.
- Reenvía el DAO recibido a su padre preferido para asegurar la propagación del destino anunciado hacia el nodo raíz.

De esta manera, en el modo storing, las rutas descendentes se encaminarán salto a salto. En el modo non-storing, el mensaje DAO, además del prefijo del destino anunciado, también contiene la dirección de su padre preferido. En este caso, sin embargo, cuando los nodos reciben un mensaje DAO, lo reenvían directamente a su padre preferido sin almacenar ninguna información hasta que finalmente lo recibe el nodo raíz del DODAG. Una vez el nodo raíz recibe el mensaje DAO transmitido, almacena la información recibida en su tabla de encaminamiento en forma de una relación padre-hijo, que se usará posteriormente para construir en el origen (nodo raíz) la ruta hasta el destino. De este modo, cuando el nodo raíz necesite comunicarse con un destino específico, adjunta la ruta desde el origen

hasta el destino en la cabecera del paquete y lo envía al siguiente salto. Cuando un nodo intermedio recibe este paquete, simplemente inspecciona su cabecera para determinar a qué nodo debería reenviar el paquete.

RPL también proporciona soporte para tráfico P2P para que dos nodos dentro de la red puedan comunicarse entre ellos. Para ello, cuando un nodo necesita enviar un paquete a otro nodo dentro de un DODAG, primero el paquete es enviado a través de las rutas ascendentes del DODAG hasta que llega a un nodo antepasado común entre los dos nodos. Y después, el paquete es reenviado a través las rutas descendentes del DODAG desde este nodo antepasado hasta el nodo destino.

En la Figura 2.2c se puede ver el intercambio de mensajes DAO para formar las rutas descendentes, mientras que en la Figura 2.2d se ilustra el estado final del DODAG, con las rutas ascendentes y descendentes ya construidas.

2.2.2 Funciones objetivo (OF)

Con el fin de poder adaptarse a todos los requisitos conflictivos de las diferentes aplicaciones de las redes de sensores, RPL desacopla la selección de ruta y mecanismos de optimización de las operaciones del núcleo del protocolo como el procesado y reenvío de paquetes. Por lo tanto, el núcleo del protocolo está centrado en la intersección de esos requisitos, mientras que los módulos adicionales se diseñan para objetivos específicos como minimizar la energía consumida o maximizar la fiabilidad [6].

El termino Función Objetivo (objective function, OF) se usa para describir el conjunto de reglas y políticas que gobiernan los procesos de selección y optimización de ruta de manera que pueda satisfacer los diferentes requisitos de varias aplicaciones. En términos técnicos, la función objetivo se utiliza para dos objetivos principales:

- Primero, especifica cómo se obtiene el rango dependiendo de una o varias métricas de encaminamiento (por ejemplo, energía, número de saltos, latencia, rendimiento, calidad del enlace y color del enlace).
- Y segundo, define cómo se debe usar el rango para seleccionar al padre preferido.

Actualmente se han estandarizado dos funciones objetivo para RPL, la Función Objetivo Cero (Objective Function Zero, $OF0$) y la Función Objetivo de Mínimo Rango con Histéresis (Minimum Rank with Hysteresis Objective Function, $MRHOF$). $MRHOF$ puede producir resultados más estables, pero $OF0$ es más simple y converge más rápidamente [26].

Objective Function Zero (*OF0*)

La $OF0$ [27] está diseñada para seleccionar el nodo más cercano al nodo raíz del DODAG como el padre preferido. El rango de un nodo (Rn) es calculado sumando un valor escalar estrictamente positivo (*rank_increase*) al rango de un padre preferido seleccionado (Rp) como se muestra en las ecuaciones 2.1, donde step-of-rank (Sp) representa un valor relacionado con la métrica de enlace y las propiedades del padre, como el ETX

(contador de transmisiones esperadas) definido en [28], mientras el factor de rango (Rf) y *stretch_of_rank* (*Sr*) son factores normalizados.

$$\begin{aligned} Rn &= Rp + rank\_increase \\ rank\_increase &= (Rf * Sp + Sr) * MinHopRankIncrease \end{aligned} \tag{2.1}$$

La $OF0$ no especifica qué métrica o métricas deberían estar involucradas en el cálculo del incremento de rango. Para la selección del padre, un nodo ejecutando $OF0$ considera siempre padre preferido a aquel que haga que *Rn*, el rango propio, sea menor. $OF0$ considera también seleccionar a otro padre como reserva en caso de que la comunicación con el padre preferido se pierda.

Minimum Rank with Hysteresis Objective Function (*MRHOF*)

La $MRHOF$ [29] está diseñada con el objetivo de evitar excesivos cambios en la topología de la red (es decir, cambios frecuentes de padre preferido). En la $MRHOF$, un nodo calcula el coste de ruta hasta el nodo raíz (path cost) a través de cada vecino sumando dos componentes: el valor de la métrica de enlace o nodo con el vecino candidato y el valor de la métrica anunciada en el Metric Container (campo de los mensajes DIO). Después de calcular el coste de ruta a través de todos los padres candidatos, un nodo selecciona al padre con menor coste de ruta como padre preferido. Sin embargo, a diferencia de $OF0$, $MRHOF$ cambiará de padre preferido si la diferencia entre el nuevo coste de ruta mínimo calculado y el coste de ruta del padre preferido es mayor que PARENT_SWITCH_THRESHOLD, que es el valor de la histéresis en $MRHOF$. Si hay múltiples padres candidatos que comparten el mismo coste de ruta, se podría usar otra métrica para el desempate, como energía restante en el nodo.

2.2.3 Mantenimiento de encaminamiento (*Trickle Timer*)

Uno de los principios clave en el diseño de RPL es minimizar la sobrecarga del control de encaminamiento y el coste de la señalización para reducir la energía consumida y mejorar la fiabilidad. Con este objetivo, RPL implementa el algoritmo de goteo ($Trickle$) [30], con el que controla el tráfico de señalización utilizado para construir y mantener el DODAG.

La idea básica detrás de $Trickle$ es ajustar la frecuencia de transmisión de mensajes basándose en las condiciones de la red. $Trickle$ depende de dos mecanismos simples para diseminar información de encaminamiento eficientemente:

- El primer mecanismo consiste en cambiar adaptativamente la ratio de señalización acorde con las condiciones actuales presentes en la red. Específicamente, $Trickle$ incrementa la ratio de transmisión cuando se descubre un cambio en la información de encaminamiento (es decir, se detecta una incongruencia), como medio para poblar la red rápidamente con información actualizada. A medida que la red se estabiliza, $Trickle$ reduce exponencialmente la ratio de transmisión para limitar el número de transmisiones cuando no hay información actualizada que propagar.
- El segundo mecanismo usado por $Trickle$ es el mecanismo de supresión, con el cual, el nodo suprime la transmisión de su paquete de control si detecta que ha

habido suficientes vecinos que han trasmitido la misma información, limitando así las transmisiones redundantes.

Se han propuesto mejoras al algoritmo de *Trickle*, como en [31], pero la última estandarización por parte del IETF [30] es del año 2011.

2.2.4 Mecanismos de regeneración de RPL

Para protegerse de los fallos en la red, RPL proporciona mecanismos de regeneración para detectar y eliminar bucles y reparar el DODAG. En los siguientes apartados se verán ambos mecanismos.

Detección y eliminación de bucles

En los protocolos de vector-distancia, incluido RPL, los bucles son un problema común que puede deberse a múltiples razones (por ejemplo, pérdida de paquetes de control) y afectar negativamente al funcionamiento de la red. El encaminamiento basado en rango usado por RPL sirve como mecanismo para la eliminación de bucles, mientras que para su detección usa otro mecanismo simple llamado validación de ruta de datos.

En la validación de ruta de datos, RPL introduce información de encaminamiento en los paquetes de datos transmitidos que indican la dirección del flujo (es decir, ascendente o descendente) y el rango del emisor del paquete. Por lo tanto, una incoherencia entre el nodo emisor y receptor es indicativo de un posible bucle. Por ejemplo, si un nodo recibe un paquete moviéndose en sentido descendente de un nodo que tiene mayor rango, entonces el nodo receptor puede deducir que ha ocurrido una incoherencia, ya que un paquete recibido de un nodo con mayor rango sólo puede llevar dirección ascendente.

Reparación del DODAG

Para superar los fallos que puedan ocurrir en la red, como los bucles, RPL usa dos métodos de reparación dinámicos llamados reparación global (Global Repair) y reparación local (Local Repair).

En la reparación local (Local Repair), un nodo no raíz que detecta una incoherencia (por ejemplo, un bucle o un fallo en el enlace), debería abandonar el DODAG anunciando el rango INFINITE_RANK para envenenar sus rutas y después reengancharse al DODAG como un nodo nuevo.

En cambio, en la reparación global (Global Repair), se reconstruye totalmente la topología del DODAG. La reparación global sólo puede ser lanzada por el nodo raíz del DODAG al detectar un fallo en la red y se anuncia a los otros nodos incrementando el campo de versión del DODAG dentro de los mensajes DIO.

2.2.5 Seguridad en RPL

La seguridad de RPL, por lo general, se basa en mecanismos de capa de enlace para soportar las características de seguridad de autenticidad, integridad y confidencialidad. Sin embargo, RPL puede usar sus propios mecanismos de seguridad con tres modos de seguridad opcionales definidos en el estándar como se indica a continuación:

1. Modo inseguro: En este modo, los mensajes de control de RPL son transmitidos sin incluir ninguna característica adicional de seguridad. En este caso, RPL depende de

las primitivas de seguridad de otras capas para satisfacer los requisitos de seguridad de la red.

2. Modo de seguridad preinstalado: En el modo preinstalado, los nodos cuentan con claves preinstaladas con las que se pueden generar y procesar mensajes RPL seguros.
3. Modo de seguridad autenticado: Como el modo preinstalado, los nodos cuentan con claves preinstaladas. Sin embargo, solo pueden ser utilizadas para que los nodos puedan unirse a la instancia RPL como hojas (nodos que no reenvían mensajes de otros nodos). Si lo nodos quieren unirse como encaminadores, se les exigirá otra clave de una autoridad de autenticación.

2.2.6 Implementaciones más extendidas

Podemos encontrar distintas implementaciones del protocolo RPL con mayor o menor complejidad [25, 32]. Se pueden encontrar implementaciones de RPL en los siguientes sistemas operativos [33, 34]:

- Contiki-NG
- TinyOS
- EyeOS
- RIOT
- LiteOS
- T-Kernel

Contiki y TinyOS son los sistemas operativos más usados en las investigaciones sobre RPL, sobrepasando el 90% [35]. También existen implementaciones en los siguientes simuladores:

- Omnet++
- NS-3
- Cooja

Contiki-NG [36] es un sistema operativo que tiene una implementación de RPL llamada ContikiRPL. Contiki-NG surgió en 2017 como una evolución del sistema operativo Contiki [37]. Contiki-NG contiene dos implementaciones de ContikiRPL: ContikiRPL-Classic y ContikiRPL-Lite.

ContikiRPL-Classic es la continuación de la implementación RPL original de Contiki: ContikiRPL. Esta implementación se creó ya en 2009, mientras el estándar RPL aún estaba en desarrollo. A lo largo de los años, se han agregado muchas funciones al RFC de RPL y a los RFCs relacionados, como soporte para múltiples instancias y múltiples DODAG, modo almacenamiento (storing) y no almacenamiento (non-storing), multidifusión (multicasting) y más. Por lo tanto, al precio de admitir una gran cantidad de funcionalidades de estándares y borradores de Internet, la implementación se ha vuelto compleja, ha adquirido una gran huella de ROM y, por lo tanto no es adecuada para motas con recursos limitados.

ContikiRPL-Lite es la implementación RPL predeterminada de Contiki-NG. Comenzó como una reescritura importante de la versión en 2017 de ContikiRPL, con un enfoque en la funcionalidades más importantes y estables, siguiendo a las que la comunidad ha usado en muchas implementaciones y experimentos de investigación. ContikiRPL-Lite elimina la compatibilidad con el modo almacenamiento (storing mode) en favor del modo sin almacenamiento (non-storing mode) y elimina la complejidad de manejar múltiples instancias y DODAG. A través de estos cambios, ContikiRPL-Lite normalmente muestra un mejor rendimiento y tiene una huella ROM considerablemente menor, por lo que es más adecuado para motas con mayores limitaciones de memoria. Por otro lado, estas optimizaciones tienen un nivel de interoperabilidad más bajo con otras implementaciones, que pueden usar algunas funcionalidades no implementadas por ésta, como el modo almacenamiento, por ejemplo. En la tabla 2.1 se reflejan las principales diferencias de funcionalidad entre ContikiRPL-Classic y ContikiRPL-Lite.

TinyOS es un sistema operativo que tiene una implementación de RPL llamada TinyRPL. Fue creado en 2002 por un consorcio liderado inicialmente por la Universidad de California en Berkeley y posteriormente por la universidad de Stanford [38] en el que también se encontraban empresas como Intel. TinyOS es un sistema operativo controlado por eventos y su principal ventaja es que está diseñado para nodos de redes de sensores que tienen recursos muy limitados (por ejemplo, 8K bytes de memoria de programa, 512 bytes de RAM). Una de sus limitaciones es el uso de un dialecto del lenguaje de programación C llamada nesC [39] (pronunciado "NES-see") diseñado para incorporar los conceptos de estructuración y el modelo de ejecución de TinyOS. Las implementaciones de RPL en TinyOS [40] se centran en dos capas de la pila de software, BLIP, la pila IP de bajo consumo de Berkeley y TinyRPL, la propia implementación de RPL. BLIP es la pila IPv6/6LoWPAN de facto para TinyOS, y la implementación de RPL en TinyOS interactúa con las interfaces que proporciona BLIP. BLIP implementa la compresión de encabezados 6LoWPAN, el descubrimiento de vecinos y DHCPv6 para permitir el uso de IPv6 por las capas superiores. Actualmente TinyOS es un proyecto discontinuado, y aunque existen grupos de trabajo externos a la asociación original que han seguido desarrollando el proyecto de TinyOS en un *fork* llamado TinyProd [41], éste también parece discontinuado, ya que no ha tenido actividad en los últimos dos años [42].

Entre los simuladores disponibles, podemos mencionar Omnet++ y NS-3 [43]. Este último es un simulador de red de eventos discretos ampliamente utilizado, destinado principalmente a uso educativo y de investigación. Algunos autores propusieron modelar RPL en NS-3 [44] y otros llegaron a publicar que habían implementado una versión muy básica de RPL [32]. Sin embargo, en la web oficial de NS-3 [45], se indica que a partir de la versión 3.19 (enero de 2014) empezó a plantearse el desarrollo limitado de RPL dentro del modelo lr-wpan (Low-Rate Wireless Personal Area Network), del que ya estaban parcialmente implementados los niveles físico, de acceso al medio, y 6LoWPAN [46]. Ese desarrollo nunca se ha incorporado a las versiones publicadas, y a partir de 2019 se comenzó a trabajar en una refactorización del código, sin resultados aparentes [47].

Cooja [48] es un simulador que permite simular motas con el sistema operativo Contiki-NG, y por lo tanto se pueden probar distintos aspectos de RPL, tanto ContikiRPL-Classic como ContikiRPL-Lite. El simulador Cooja está escrito en Java, lo que permite ejecutarlo en cualquier entorno que tenga la máquina virtual de Java, lo cual es una gran ventaja. El

Tabla 2.1 Características de las distintas implementaciones de ContikiRPL.

	ContikiRPL-Classic	ContikiRpl-Lite
rutas ascendentes	X	X
rutas descendentes	X	
múltiples instancias	X	
múltiples DODAG	X	
modo storing	X	
modo non-storing	X	X

código que simula es el del propio sistema operativo Contiki-NG, que está escrito en C, con lo que cualquier modificación que se quiera hacer es usando este lenguaje, que es un estándar. Un posible inconveniente del simulador es que para ejecutar un script al final de la simulación hay que usar el lenguaje JavaScript (formalmente ECMAScript [49]), un lenguaje interpretado que se ejecuta al final de la simulación y que si contiene algún error no se detecta hasta el final de ésta. Este inconveniente se puede soslayar utilizando el script únicamente para obtener los resultados de la simulación y posteriormente procesarlos con scripts en otro lenguaje como Python, que tiene bibliotecas adecuadas para ello, una vez terminada la simulación.

En esta tesis se va a usar RPL Lite, en motas con el sistema operativo Contiki-NG, realizando simulaciones en el simulador Cooja, debido a su integración nativa con el entorno de desarrollo (Cooja es un subproyecto dentro del proyecto Contiki-NG, accesible directamente en la carpeta *tools*).

2.2.7 Capa cruzada (cross-layer)

Teniendo en cuenta que RPL es un protocolo de encaminamiento, y es por lo tanto un protocolo de la capa 3 del modelo OSI [50], lo ortodoxo es usar información obtenida a través de esa capa y actuar sólo sobre parámetros y mecanismos presentes en esa capa.

En el enfoque de capa cruzada (cross-layer), se usa información obtenida en distintas capas de comunicación, y se actúa también sobre esas capas. En concreto se interactúan con la capa de encaminamiento (capa 3), capa de nivel de enlace (capa 2) y capa física (capa 1).

2.3 Propuestas de trabajos relacionados

Esta tesis busca mejorar la eficiencia energética en redes de sensores inalámbricos que usan en los niveles inferiores el estándar IEEE 802.15.4 (la elección del IETF para IoT), lo cual no es nuevo. Ese ahorro de energía se busca principalmente en las comunicaciones, en la interfaz radio, ya que es la principal contribuyente al consumo de energía [51]. Para minimizar el consumo de energía en las comunicaciones, se pueden aplicar mecanismos en las distintas capas del modelo OSI, desde la capa física, hasta la capa de red, pasando por la capa de enlace.

2.3.1 Duty Cycling

Un primer mecanismo y el más directo es mantener apagada la interfaz radio el mayor tiempo posible, y en ello se basan los mecanismos de *Duty Cycling* [52] o *Ciclo de trabajo*, que consisten en encender y apagar la interfaz radio para conservar la energía, y que se corresponden con la capa física y la subcapa MAC de la capa de enlace. El objetivo principal del ciclo de trabajo es reducir el consumo de energía de las motas y, como consecuencia, aumentar la longevidad general de la red. Más específicamente, el ciclo de trabajo tiene como objetivo reducir la escucha inactiva, es decir, hacer que el transceptor de radio espere en vano una trama. La dificultad de mantener la radio encendida sólo cuando es necesario es que, para la mayoría de las aplicaciones, las motas no saben de antemano cuándo van a llegar los datos. La escucha inactiva no es la única fuente de desperdicio de energía. La sobre-escucha (cuando un nodo desperdicia energía escuchando tramas que no interesan), la sobrecarga de paquetes de control y las colisiones, también desperdician energía. Es importante identificar estas causas ya que, al intentar reducir la escucha inactiva, el ciclo de trabajo puede aumentar las tasas de colisión e introducir un mayor tráfico de control, efectos secundarios que pueden aumentar el consumo de energía que el ciclo de trabajo está tratando de reducir. El estándar IEEE 802.15.4 introduce un mecanismo de ciclo de trabajo estructurado en tramas baliza que no es muy efectivo para la topología en malla, una topología que es muy común en las WSN. Por este motivo se han creado más mecanismos de este tipo que sean efectivos con esta topología y así mejorar el ahorro energético de los nodos. Estos mecanismos se pueden clasificar en dos técnicas diferentes, programación y escucha muestreada. En la técnica de programación, utilizada por TSCH, los nodos se sincronizan y guardan información sobre los vecinos acerca de cuándo van a estar escuchando, y la utilizan para saber cuándo transmitir. Mientras que en la técnica de escucha muestreada, utilizada por ContikiMAC, el canal es monitorizado periódicamente para determinar si alguna trama está siendo transmitida por un vecino.

TSCH

TSCH (Time Slotted Channel Hopping) es un modo de funcionamiento de ciclo de trabajo del estándar IEEE 802.15.4e [53] que fue creado para actualizar y mejorar el estándar IEEE 802.15.4 y adaptarlo para determinadas aplicaciones, sobre todo del campo de la industria. TSCH gestiona las transmisiones y recepciones de todos los nodos de la red de manera síncrona, siendo de especial interéspara aplicaciones con requisitos estrictos de tiempo, ahorro energético y fiabilidad de la red. TSCH combina el acceso al canal por intervalos de tiempo, con acceso multicanal y un mecanismo de salto de canal. Los beneficios de estas características son los siguientes:

- El acceso al canal por intervalos de tiempo incrementa el rendimiento de la red eliminando las colisiones entre los nodos competidores y ofreciendo latencias deterministas a la aplicación.
- La comunicación multicanal permite a varios nodos intercambiar tramas al mismo tiempo usando diferentes canales. Esto hace que la capacidad de la red aumente.
- El mecanismo de salto de canal permite mitigar los efectos de las interferencias aumentando la fiabilidad de la comunicación.

Dentro de un intervalo de tiempo, el receptor espera recibir el paquete exactamente un determinado tiempo después del inicio del intervalo de tiempo, y lo mismo con el asentimiento. Por este motivo, este mecanismo necesita de una buena sincronización para que funcione correctamente, y debido a diferencias en producción o temperaturas, la frecuencia de reloj de los nodos tiene ligeras diferencias que provocan problemas de sincronización. Para corregir esas pequeñas diferencias de la frecuencia de reloj, el receptor empieza a escuchar el canal un tiempo (tiempo de guarda) antes, pero, aun así, el sistema requiere de una resincronización periódica para un correcto funcionamiento. Para esta resincronización, cada nodo tomará como referencia a un vecino, y cuando reciba un paquete o ACK de éste, anotará el instante de inicio de la recepción y le restará el tiempo determinado para conocer el tiempo de referencia, con el cual sincronizará su reloj. Esta necesidad de sincronización hace que TSCH sea más difícil de implementar que otros mecanismos de ciclo de trabajo.

ContikiMAC

ContikiMAC [54] es un protocolo que se encarga de la transmisión de las tramas MAC actuando directamente sobre la capa física. Para ello, trata de gestionar el encendido de la radio despertándola periódicamente para escuchar el canal y comprobar si los vecinos están transmitiendo. Cuando un nodo detecta que un paquete se está transmitiendo durante una de las comprobaciones periódicas, se mantiene despierto hasta que recibe un paquete completo y envía el ACK. Para enviar un paquete, el emisor lo envía repetidas veces hasta que recibe el ACK del receptor. En el caso del envío a difusión o broadcast, como no se espera la recepción de un ACK, se enviará el paquete repetidamente durante un intervalo de tiempo suficiente que garantice la detección por parte de los receptores. Durante los periodos donde la radio despierta para ver si se está transmitiendo un paquete, en realidad se está midiendo el RSSI (Received Signal Strength Indicator) del canal. Si el RSSI supera un determinado umbral indica que se podría estar transmitiendo un paquete, pero también podría ser ruido o un paquete con otro destino. Para diferenciar estos estados ContikiMAC tiene un sistema de detección de paquetes que juega con los tiempos entre paquetes y la duración máxima de transmisión de un paquete para comprobar si se está recibiendo uno, o ir a dormir en el caso de que no. ContikiMAC presenta un mecanismo denominado *Transsmission Phase-Lock*, que trata de minimizar el envío repetido hasta recibir el ACK. Este mecanismo se basa en que el receptor se va a despertar de forma periódica para comprobar si tiene que recibir un paquete, de manera que el transmisor puede aprender cuándo se despierta y enviar el paquete justo antes. Este aprendizaje viene cuando, para un receptor determinado, ya se ha enviado un paquete con éxito porque se ha recibido su ACK, pudiendo calcular así cuándo podría estar el receptor despierto. En conclusión, ContikiMAC permite ahorrar energía apagando la radio cuando no se está transmitiendo y encendiéndola periódicamente para permitir la recepción, pero este ahorro se verá afectado por la necesidad de enviar varias veces un mismo paquete hasta que se reciba. La solución que presenta ContikiMAC es el mecanismo *Transsmission Phase-Lock* que intenta que el número de reenvíos del mismo paquete sea el mínimo.

2.3.2 Tasa de transmisión adaptativa

Para minimizar el consumo de energía también se han propuesto soluciones que tienen en cuenta el encaminamiento y la tasa de transmisión (Line Rate Adpatation algorithms), como en [55] Passos et al. o también en [56, 57] Narayan et al., donde la tasa de transmisión se ajusta teniendo en cuenta el encaminamiento para minimizar el consumo de energía. Estos trabajos se aplican a redes inalámbricas en las que se puede adaptar la tasa de transmisión, como WiFi, basadas en la norma IEEE 802.11. Pero las redes basadas en la norma IEEE 802.15.4, que son las que se estudian en esta tesis, están diseñadas para redes de baja potencia y de baja tasa que producen paquetes de pequeño tamaño y operan a una tasa fija de 250 Kb/s en la banda ISM de 2,4GHz (aunque también se han definido variantes adaptadas a regulaciones regionales o aplicaciones específicas [58] Alkama et al.). Y aunque algunos autores como Jang et al. [59] han explorado esquemas de adaptación de tasa de enlace dinámica, no está soportado por la norma IEEE 802.15.4. En esta tesis se usa la potencia de transmisión como variable de control de la capa física, en lugar de la tasa de transmisión.

2.3.3 Ahorro de energía en RPL

RPL ya cuenta con varias mejoras orientadas a mejorar la eficiencia energética, incluido el balanceo de carga [60, 61], múltiples rutas [62, 63] y todo tipo de métricas combinadas, que generalmente incluyen una métrica de calidad del enlace, como el recuento de transmisiones esperadas (ETX) [64, 65] más algún término relacionado con la energía (por ejemplo, normalmente la energía residual del nodo) [66, 67]. Sin embargo, a pesar del intenso esfuerzo de investigación, mejorar la eficiencia energética persiste como un desafío [68, 66, 69]. En opinión del autor, dos aspectos contribuyen a la complejidad de este problema. En primer lugar, la optimización de la eficiencia energética es un término amplio que puede referirse a diferentes objetivos, como minimizar la energía total consumida por la red o maximizar la energía residual de los nodos (es decir, la vida útil de la red) [64, 67, 70], y optimizar uno puede evitar la optimización del otro. Por ejemplo, minimizar la energía total consumida por la red puede dar como resultado una ruta óptima en la que los nodos con una alta tasa de entrega agotan su batería más rápido que el resto y, en consecuencia, no logran alcanzar la máxima vida útil posible de la red. En segundo lugar, y siguiendo el ejemplo anterior, suele existir un equilibrio entre la optimización de la energía y la calidad del servicio (QoS). Para hacer frente a requisitos contradictorios en la misma aplicación, los autores suelen sugerir el uso de métricas de encaminamiento compuestas [71, 72, 73].

Esta tesis, sin embargo, no se centra exclusivamente en la función de encaminamiento sino también en la configuración de la potencia de transmisión de los nodos. En una implementación típica, todos los nodos funcionan con una configuración de potencia de transmisión fija. Esta suposición se hace habitualmente en la literatura existente, aunque no es raro que los nodos sensores puedan utilizar diferentes niveles de potencia de transmisión. Por ejemplo, el transceptor CC2420 utilizado por las motas Crossbow TelosB y Zolertia Z1 cuenta con ocho niveles de potencia diferentes que van desde $-25dBm$ a $0dBm$, el transceptor TI CC1200 utilizado por Zolertia Re-Mote cuenta con 31 niveles de potencia diferentes que van desde $-16dBm$ a $14dBm$ (en pasos de $1dBm$). Tener varios niveles de potencia de transmisión aporta un nuevo grado de libertad al problema de encaminamiento.

La potencia de transmisión afecta el alcance del nodo (es decir, la capacidad de ser visto como un vecino) y las estadísticas del enlace (por ejemplo, RSSI y ETX observados por los vecinos), lo que influye en la topología de encaminamiento creada. Se aprovecha este hecho para proponer un mecanismo que opere en tres capas (capa cruzada): red, enlace de datos y física.

Este enfoque de capa cruzada (cross-layer) ya se ha estudiado con anterioridad, y se pueden encontrar en la literatura distintas propuestas, aunque ninguna con el enfoque dado en esta tesis.

Un caso de capa cruzada, entre la capa de red y la subcapa MAC, llamado RPL-SCSP, lo ha propuesto Abdessalem et al. [74], que combina el ETX y la carga de la cola (Queue Load) para optimizar el retardo extremo a extremo y el consumo de energía, pero difiere de esta tesis en que, aunque su objetivo es proporcionar una cierta calidad de servicio, no se tiene en cuenta la potencia de transmisión de los nodos.

También hay enfoques de capa cruzada entre el nivel de enlace y RPL, como el propuesto por Safaei et al. ELITE [75], en el que se usa una métrica de encaminamiento que tiene en cuenta información del ciclo de trabajo de la radio (radio duty cicle). Este esquema alcanza una reducción significativa del consumo de energía, pero está restringido a protocolos que implementan el ciclo de trabajo de la radio, como ContikiMAC, y además no tiene en cuenta los nodos que pueden transmitir a diferentes niveles de potencia.

En la mayoría de la literatura relacionada con el encaminamiento en el IoT se asume que todos los nodos transmiten a un nivel de potencia por defecto. Asumir lo contrario da lugar a un nuevo problema (potencia óptima de transmisión), que está entrelazado con el problema de encaminamiento, ya que la potencia de transmisión de cada nodo afecta a la cobertura y a la calidad del enlace (por ejemplo, retraso y pérdidas). Hasta la fecha, pocos trabajos han abordado este problema.

En [76] Chipara et al. proponen un protocolo de encaminamiento en el que cada nodo ajusta su nivel de potencia de transmisión para minimizar el retraso de comunicación en tiempo real para un patrón de generación de datos conocido. Los paquetes que no tienen una fecha límite urgente se transmiten con menor potencia para reducir el consumo de energía. Sin embargo, este trabajo es diferente al propuesto en esta tesis en los siguientes aspectos: (a) crea un nuevo protocolo de encaminamiento en lugar de usar RPL estándar, (b) el criterio que se utiliza para establecer el nivel de potencia de transmisión del nodo está relacionado con el retraso y no con la energía, (c) requiere tener conocimiento previo de la disposición de los nodos y del patrón de tráfico de la capa de aplicación.

En [77] Rukpakavong et al. proponen, como se hace en esta tesis, la configuración automática de la potencia de transmisión de los nodos empleando modificaciones de RPL. Sin embargo, su trabajo difiere de esta tesis en varios aspectos. En primer lugar, sólo consideran estadísticas de enlace para la potencia de transmisión máxima, mientras que en esta tesis se mantienen estadísticas de enlace para cada nivel de potencia de transmisión disponible. En segundo lugar, en su trabajo, RPL selecciona la mejor ruta basándose únicamente en ETX, ignorando la potencia de transmisión de los nodos. Como tal, la ruta óptima es idéntica a la de función objetivo por defecto ($OF0$). Una vez configurada la topología, cada nodo ajusta automáticamente su potencia de transmisión utilizando la intensidad de la señal recibida de los mensajes DIO de los padres. Por lo tanto, no es un esquema de capa cruzada, ya que la función objetivo de encaminamiento es independiente

y desconoce la potencia de transmisión de los nodos. Sin embargo, en esta tesis, la métrica de encaminamiento tiene en cuenta tanto ETX como la potencia de transmisión del nodo (lo que podría conducir a soluciones de encaminamiento diferentes a las de $OF0$), y es la función de encaminamiento la que elige la potencia de transmisión óptima para el nodo.

En [78] Kim et al. proponen ajustar la potencia de transmisión para maximizar el rendimiento, pero no para minimizar energía, mejorando el balanceo y el problema del terminal oculto. En este trabajo se quiere mejorar la solución a dos problemas: el balanceo (load imbalance) y el terminal oculto (hidden terminal). Cuando se trasmite a $0dBm$ (máxima potencia) hay pérdidas en el enlace por el problema de terminal oculto, pero no por sobrecarga en las colas. Al bajar la potencia hay menos pérdidas en el enlace, pero empieza a haber pérdidas por sobrecarga, ya que algunos nodos tienen muchos descendientes. Su objetivo es maximizar el rendimiento (throughput), pero no minimizar la energía. Envían periódicamente paquetes DIO a máxima potencia y miden el RSSI de la recepción de esos paquetes. Con el RSSI calculan unos umbrales que afectan a la hora de decidir la topología y la potencia de transmisión, por eso lo llaman PC (power controlled) Joint control. Una diferencia importante con esta tesis es que no utilizan mensajes de sondeo (probing), alegando que es más adecuado para entornos más dinámicos. En esta tesis sí se utilizan mensajes de sondeo, que permiten mantener frescas las estadísticas del enlace con los vecinos y para distintas potencias.

En [79] Qolami et al. tienen en cuenta la calidad del enlace para ajustar la potencia de transmisión. En este trabajo cada nodo ajusta la potencia de transmisión de cada paquete y de cada asentimiento (ACK), basándose en la medición de la calidad del enlace. Se utiliza el cociente entre la potencia transmitida y la potencia recibida con cada vecino para así calcular la potencia mínima a la que se puede transmitir para que se reciba a una potencia umbral, P_{thr}, a la cual la señal recibida pueda ser decodificada correctamente. Utiliza el modo baliza-habilitada (beacon-enabled) de la norma IEEE 802.15.4 para ahorrar más energía, y tanto los paquetes de control como las balizas (beacons) se transmiten siempre con potencia máxima. Los nodos hijo, después de recibir las balizas de los posibles padres, escogen el padre preferido y utilizan el cociente entre la potencia a la que se transmitió la baliza (potencia máxima) y la potencia recibida para calcular la potencia a la que transmitir el primer paquete de datos. Para los siguientes paquetes de datos utiliza las potencias de los asentimientos recibidos. Este proceso se repite hasta la siguiente baliza. Los nodos padre, después de enviar una baliza a potencia máxima, entran en un bucle en el que después de recibir un paquete de datos envía el asentimiento (ACK) a una potencia calculada en función del cociente entre la potencia a la que se transmitió el paquete de datos y la potencia a la que lo ha recibido. Este proceso se repite hasta el final del periodo activo. En este trabajo la potencia a la que se transmiten las balizas es la potencia máxima, pero los paquetes de datos o de asentimiento se pueden transmitir a otra potencia. Cuando se recibe una baliza, se sabe que se transmitió a potencia máxima, pero no se especifica cómo calculan los nodos que reciben un paquete, que puede ser un paquete de datos o un asentimiento, a qué potencia se transmitió ese paquete.

En [80] Estepa et al. proponen un enfoque de RPL de capa cruzada (xRPL). En este enfoque, la función de encaminamiento gestiona el nivel de potencia de transmisión local del nodo. Para lograr esto, los nodos evalúan y realizan un seguimiento de las estadísticas de enlace para dos niveles de potencia de transmisión, alto y bajo, ambos disponibles en

el nodo. RPL funciona identificando las rutas más eficientes utilizando una métrica de encaminamiento que minimiza la cantidad de energía utilizada para la transmisión hacia el nodo raíz. Después de esto, cada nodo ajusta su potencia de transmisión al nivel alto o bajo, dependiendo del nodo padre seleccionado. En escenarios pequeños y densos, se ha demostrado que xRPL reduce significativamente el consumo de energía en comparación con el RPL predeterminado. Además, xRPL se alinea con las direcciones futuras descritas en [81], que apuntan a minimizar el consumo de energía a través de diseños entre capas (RPL e IEEE 802.15.4) que no requieren métodos de optimización complejos o excesivos mensajes de control. Desafortunadamente, en un intento por ofrecer una solución simple y reducir los mensajes de control excesivos, xRPL muestra una limitación importante: tomar una decisión binaria. Un problema importante con xRPL es que al tomar decisiones sólo considera dos niveles de potencia de transmisión disponibles, alto y bajo. Esto pasa por alto la gama completa de niveles de potencia que los dispositivos IoT pueden ofrecer y limita la capacidad de ser beneficioso en una gama más amplia de escenarios y generalizar el esquema entre capas.

2.4 Conclusión del estado del arte

Una vez comentadas las propuestas de trabajos relacionados, se observa que el ahorro energético en las redes de sensores inalámbricas no es un tema nuevo y que se puede abordar desde distintas capas del modelo OSI, como por ejemplo desde la capa de enlace con TSCH y ContikiMAC. También se puede abordar desde la capa de red, desde la capa física y hay también soluciones de capa cruzada (crosslayer). Entre las soluciones de capa cruzada destaca xRPL [80] (dos niveles de potencia), ya que al basarse en la capa de red y física es compatible con cualquier otro tipo de soluciones de la capa MAC, como TSCH. Además, el ahorro de energía conseguido redunda en un mayor tiempo de duración de las baterías de los nodos, es decir, aumenta el tiempo de supervivencia de la red. Como ya se ha indicado, el trabajar con sólo dos niveles de potencia implica limitaciones, pero ampliar xRPL a un número arbitrario de niveles de potencia plantea un desafío que conlleva riesgos, ya que al aumentar el número de niveles conduciría a una mayor complejidad, y a más mensajes de control transmitidos, recibidos y procesados. Todo esto podría agotar la energía de los dispositivos debido a un uso más intensivo del procesador y de la radio, podría ralentizar la formación de la topología y también afectar a la calidad de servicio. Sin embargo, tener una gama más amplia de niveles de potencia puede dar como resultado que más dispositivos utilicen una potencia de transmisión distinta a la máxima, lo que reduciría el gasto de energía hasta cierto punto, dependiendo del diseño físico. Esta tesis tiene como objetivo abordar el problema de usar múltiples niveles de potencia para ahorrar el consumo de energía, minimizando al mismo tiempo la generación de mensajes de control, sin ralentizar la formación de la topología y sin degradar la calidad de servicio.

3 Propuesta para el ahorro de energía en motas IoT con RPL y múltiples niveles de potencia de transmisión

Para minimizar el consumo de energía en RPL, se puede usar una función objetivo que permita ahorrar energía, pero sin degradar el funcionamiento de la red. La norma RPL [5] obliga a que todos los nodos de una red pertenecientes a la misma instancia, usen una misma función objetivo, aunque esta función objetivo no está fijada por la norma. Los nodos que ejecutan RPL pueden emplear, para el cálculo de la función objetivo, métricas definidas por el usuario para describir las propiedades de un enlace o nodo (por ejemplo, calidad del enlace, energía residual, etc.) y hacer que estas métricas estén disponibles para la selección de ruta enviando periódicamente la información correspondiente en mensajes DIO, en la opción de contenedor de métricas (MC Metric Container) [28].

La separación de función objetivo de la especificación del protocolo central permite que RPL se adapte para cumplir con los diferentes criterios de optimización requeridos por la amplia gama de implementaciones, aplicaciones y diseños de red. De hecho, existe una gran cantidad de métricas y funciones objetivas propuestas en la literatura para superar las limitaciones naturales de RPL (pueden consultarse en [67, 66]). Hay que tener en cuenta que el objetivo de esta tesis no es competir con estas métricas, sino complementarlas. De hecho, el mecanismo entre capas propuesto se puede adaptar fácilmente a otras métricas sugeridas en la literatura.

A continuación, se describirán las limitaciones de RPL cuando todos los nodos transmiten a una única potencia. Posteriormente se describirá una adaptación de RPL diseñada para reducir el consumo energético cuando las motas transmiten a distintas potencias. Finalmente se estudiará el mecanismo de Probing y se proporcionará un nuevo algoritmo que reduzca el consumo energético asociado a Probing cuando se utilizan múltiples potencias de transmisión.

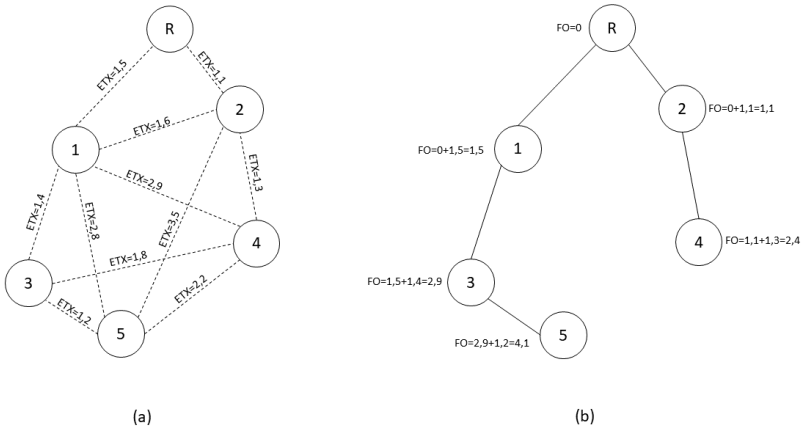


Figura 3.1 Ejemplo del cálculo de la FO a partir de los ETX.

3.1 Limitaciones de RPL con nodos de potencia única

La función objetivo por defecto de RPL, la $OF0$ [27], utiliza típicamente como métrica aditiva ETX (contador de transmisiones esperadas) definido en [28], seleccionando el camino cuya suma de los ETX de los enlaces que se recorren hasta llegar al nodo raíz sea mínima. Si el ETX de todos los enlaces fuera 1, que es el valor mínimo de ETX (quiere decir que no se esperan retransmisiones en los enlaces), entonces ese encaminamiento es equivalente a escoger el camino de menor número de saltos hasta llegar al nodo raíz. Como el ETX de todos los enlaces puede ser superior a 1, debido a que haya que retransmitir paquetes, aquellos enlaces con mayor ETX gastarán más energía. Teniendo en cuenta lo anterior, y supuesto que todos los nodos transmiten a la misma potencia, la función objetivo $OF0$ (reducir el ETX acumulado) equivale a escoger el camino de menor coste energético hasta llegar al nodo raíz.

Para el cálculo del ETX con los vecinos, se tienen en cuenta los paquetes enviados y los asentimientos recibidos, y a partir de ahí se calcula el ETX con cada vecino. Adicionalmente, se puede obtener una estimación inicial del ETX a partir del RSSI de los paquetes recibidos de un vecino al que no se han enviado paquetes.

Para ilustrar el cálculo de la función objetivo y la elección del nodo padre preferido en RPL se va a usar el ejemplo mostrado en la Figura 3.1a, donde hay 5 nodos más el raíz, y las líneas de puntos indican los enlaces con los nodos que son vecinos (aquellos de los que se ha recibido algún paquete). Junto a cada enlace está representado el ETX con el vecino correspondiente. En los enlaces con mayor ETX se gastará más energía, y en los enlaces con ETX de valor 1 (ETX mínimo) se gastará la mínima energía.

En la figura 3.1b se muestra el árbol DODAG que se ha escogido mediante el protocolo RPL, usando la función objetivo que escoge el camino cuya suma de los ETX de los

enlaces que se recorren hasta llegar al nodo raíz es mínima. La función objetivo usada se calcula con la fórmula (3.1).

$$\text{mín}\left(\sum_{j\in \textit{camino al raíz}} \text{ETX}_j\right) \tag{3.1}$$

Para mostrar cómo es el cálculo en un nodo, tomemos como ejemplo el nodo 5. Si el valor de todos los ETX fuera 1, el nodo 5 escogería como padre preferido al nodo 1 o al nodo 2, ya que en ambos casos la suma de los ETX sería 2 (que coincide con el número de saltos). En cambio, teniendo en cuenta los ETX que se observan en la Figura 3.1a, el nodo 5 escoge como nodo padre preferido al nodo 3, ya que la función objetivo a través de él, es decir la suma de los ETX a través de él, es menor. Para calcular la función objetivo para cada uno de los distintos vecinos y escoger como nodo padre el que consigue el valor mínimo, hay que sumar dos términos: el ETX al vecino y la suma de los ETX del camino del vecino hasta el raíz, tal como se muestra en la expresión (3.2).

$$\begin{aligned}\sum_{\substack{j\in \textit{camino}\\ \textit{al raíz}}} \text{ETX}_j &= \text{ETX}_{\textit{al vecino}} + \sum_{\substack{k\in \textit{desde}\\ \textit{vecino al raíz}}} \text{ETX}_k \\ &= \text{ETX}_{\textit{al vecino}} + \text{MC}_{\substack{\textit{recibido}\\ \textit{del vecino}}}\end{aligned} \tag{3.2}$$

El término ETX al vecino lo calcula localmente cada nodo para cada vecino, y lo tiene almacenado en la tabla de estadísticas del nivel de enlace (junto con el RSSI). Esta tabla se va actualizando cuando un nodo envía o recibe paquetes de los vecinos. El término de la suma de los ETX del camino del vecino hasta el raíz se obtiene de los mensajes DIO recibidos de los vecinos, en concreto de la opción contenedor de métrica (MC Metric Container) definida en [28], y se almacena en la tabla de vecinos RPL.

En la Figura 3.2, se puede observar el contenido de la tabla de estadísticas del nivel de enlace (subcapa MAC) y el de la tabla de vecinos RPL para el nodo 5. En la tabla de estadísticas del enlace se tiene calculado el ETX para los 4 vecinos del nodo 5. En la tabla de vecinos RPL se tiene almacenado el MC (Metric Container) recibido en los mensajes DIO de cada vecino, que es la suma de los ETX del mejor camino del vecino hasta el raíz. Este valor lo habrán calculado los nodos vecinos como el valor mínimo de su función objetivo, y lo habrán enviado en los mensajes DIO periódicos, en la opción MC (Metric Container). En esa tabla de vecinos RPL, la función objetivo (FO) para cada vecino se calcula sumando el MC con el ETX (de la tabla de estadísticas del enlace), y el vecino que tenga el menor valor en FO se escogerá como padre preferido. En el caso del nodo 5 el nodo padre preferido escogido es el nodo 3, ya que para ese vecino el MC recibido es 2,9, el ETX es 1,2 y la suma 4,1 es el menor valor. Este padre preferido se tomará como siguiente salto (Next Hop) en la tabla de reenvío IP. Ese valor mínimo de la función objetivo es a su vez el valor que usará el nodo 5 para enviar sus mensajes DIO, dentro de la opción contenedor de métrica (MC Metric Container).

En este apartado la función objetivo ha minimizado la suma de los ETX, pero si la función objetivo minimizara la suma de los ETX previamente multiplicados por la potencia de transmisión y por el tiempo de transmisión de cada paquete, entonces la función objetivo

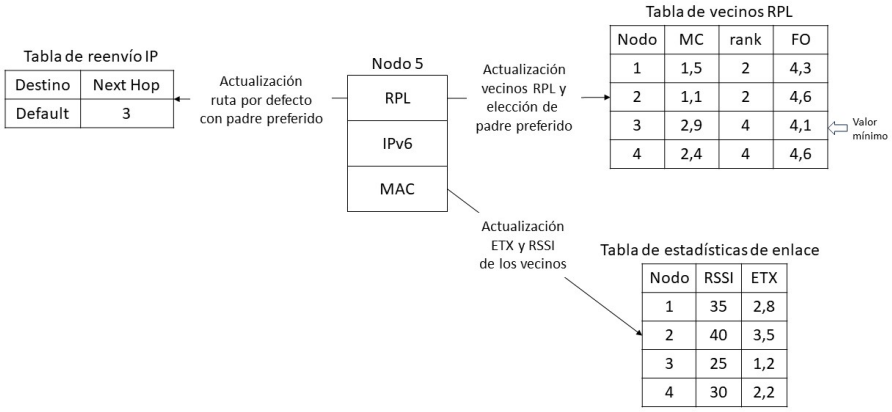


Figura 3.2 Ejemplo de tablas del nodo 5.

estaría minimizando el consumo de energía de un paquete en atravesar toda la red hasta llegar al nodo raíz, como indica la fórmula (3.3).

$$\text{mín}\,(\sum_{\substack{j\in camino\\ al\ raíz}} \text{ETX}_j \cdot \text{PTX}_j \cdot \text{T}_{paq}) = \text{mín}\,(\frac{Energía_{hasta\ raíz}}{paq}) \tag{3.3}$$

En la fórmula (3.3) PTX_j representa la potencia de transmisión del nodo y T_{paq} el tiempo de transmisión de un paquete. Teniendo en cuenta que todos los nodos transmiten a la misma potencia y a la misma tasa, y asumiendo que todos los paquetes tienen el mismo tamaño, los valores de PTX_j y T_{paq} son fijos, y por lo tanto se pueden sacar del sumatorio y no afectan a la minimización, como se muestra en la fórmula (3.4).

$$\text{mín}\,(\sum_{\substack{j\in camino\\ al\ raíz}} \text{ETX}_j \cdot \text{PTX}_j \cdot \text{T}_{paq}) = \text{PTX} \cdot \text{T} \cdot \text{mín}\,(\sum_{\substack{j\in camino\\ al\ raíz}} \text{ETX}_j) \tag{3.4}$$

En cambio, si se quiere seguir minimizando la energía, pero los nodos pueden transmitir a distintas potencias, PTX_j no se puede sacar del sumatorio, y es lo que se analiza en el siguiente apartado.

3.2 Propuesta para la adaptación de RPL a múltiples potencias de transmisión

Antes de describir el esquema propuesto, se va a hacer un resumen de la información gestionada por RPL. Tal como se ha visto en el ejemplo de la Figura 3.2, el protocolo RPL interacciona con 3 tablas:

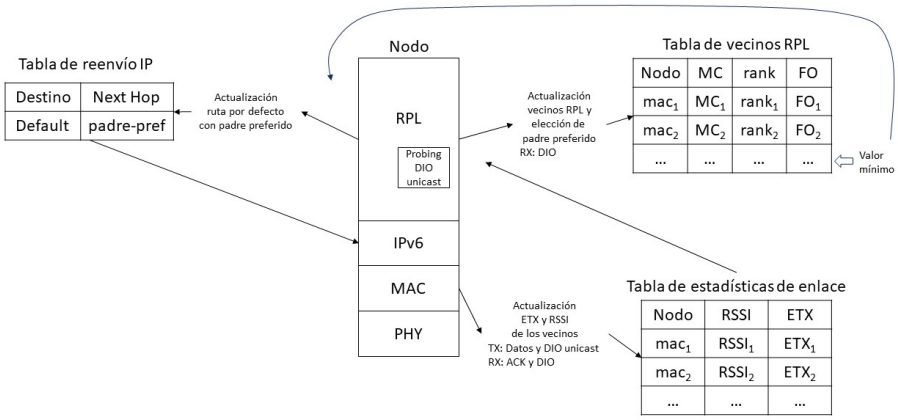


Figura 3.3 Tablas RPL y Probing.

- Tabla de reenvío IP
- Tabla de vecinos RPL
- Tabla de estadísticas de enlace

Estas tablas y su interacción con RPL se ilustra en la Figura 3.3. La tabla de reenvío IP se refiere a la tabla que se usa en IPv6 para enviar y reenviar los paquetes, que sólo contiene una entrada, ya que en esta tesis se estudia el caso de sólo tráfico ascendente, hacia el nodo raíz. Por lo tanto, la única entrada de esa tabla, la entrada que indica el siguiente salto por defecto (default) contiene la dirección del padre preferido. Esta dirección se actualiza cada vez que RPL decide cambiar de padre preferido porque haya encontrado un camino mejor, es decir, haya encontrado un nodo vecino por el que consiga disminuir la función objetivo. También es posible que se cambie de padre preferido porque ya no esté operativo (las estadísticas con ese nodo no sean frescas).

La tabla de estadísticas de enlace la mantiene actualizada la subcapa MAC tal como se indicó en el apartado 3.1. Cada vez que se recibe un mensaje de un nuevo vecino (no se conocía su dirección MAC) se crea una nueva entrada en la tabla con el RSSI del mensaje recibido y se calcula un valor inicial para el ETX. Esto ocurre por ejemplo cuando se recibe un mensaje DIO multicast de un nuevo vecino. Cuando se recibe un ACK de un mensaje enviado previamente a un vecino, sirve para actualizar el ETX correspondiente a ese vecino. En el caso de que se active el mecanismo de Probing, que es opcional, éste monitoriza la frescura (actualizaciones recientes) de las entradas de esta tabla, y hace que se envíen mensajes de control DIO unicast al vecino correspondiente, el cual enviara un ACK, lo que permitirá mantener convenientemente actualizada la tabla.

La tabla de vecinos RPL es actualizada usando la información de nivel de enlace de la tabla de estadísticas de enlace y la información recibida en los mensajes DIO. De estos

mensajes DIO, el contenido de la opción MC (Metric Container), se almacena en la tabla de vecinos RPL, en la entrada correspondiente al vecino del cual se ha recibido el DIO. La función objetivo (FO) para cada vecino (que también se almacena en esta tabla) la calcula RPL a partir del MC y del ETX (que está almacenado en la tabla de estadísticas de enlace). Cuando se producen actualizaciones, se escoge como padre preferido a aquel vecino que tiene un menor valor en la función objetivo (FO) calculada para él. La dirección de ese padre preferido es usada por RPL para actualizar la tabla de reenvío IP. Además, la información almacenada en la tabla de vecinos RPL es usada por RPL para el cálculo del rango (Rank) y la opción MC (Metric Container) que el nodo enviará en los mensajes DIO multicast periódicos.

Una vez descrito el funcionamiento de RPL, el uso del mecanismo de Probing para actualizar la tabla de estadísticas de enlace, así como la interacción con las tablas donde se almacena la información necesaria, el fundamento detrás del esquema propuesto entre capas se basa en la idea de que la potencia de transmisión de los nodos y la elección del padre preferido de RPL no son independientes. En efecto:

- La potencia de transmisión afecta al conjunto de vecinos accesibles o, de manera equivalente, al conjunto de padres RPL elegibles. La Figura 3.4 ilustra este hecho mostrando el conjunto de vecinos descubiertos por el nodo 5 como resultado de su nivel de potencia de transmisión (cuatro vecinos con el nivel máximo de potencia, tres vecinos con un nivel intermedio y dos vecinos con el nivel mínimo).
- La potencia de transmisión de un nodo puede afectar al ETX observado. Por ejemplo, en la tabla de estadísticas de enlace del nodo 5 que aparece en la Figura 3.4, el nodo 1 tiene un ETX distinto para $p = 1$ y $p = 2$. Por lo tanto, indirectamente, la potencia de transmisión también puede afectar a la selección del padre preferido en las funciones objetivo que incluyen métricas de calidad del enlace como ETX.

En general, las configuraciones de bajo consumo pueden ahorrar energía, pero podrían degradar la QoS (por ejemplo, más retransmisiones, pérdida de conectividad, rutas más largas), lo que podría aumentar el consumo de energía. Como tal, no está claro si una potencia de transmisión mayor o menor será más beneficiosa, ya que todo depende de las condiciones predominantes de la red y del diseño de los nodos. Esta relación entre las configuraciones de energía en las capas física y de red es el aspecto básico de la propuesta de esta tesis.

El hecho de tener varios niveles de potencia de transmisión disponibles genera las siguientes complicaciones potenciales:

- El mecanismo de sondeo incrementa el número de mensajes de control. El envío de mensajes DIO unicast a cada vecino y a cada nivel de potencia se traduce en que se actualizarán más objetivos. Esto podría provocar un mayor consumo de energía debido a esos mensajes de control adicionales y al procesamiento por parte de la CPU. El rendimiento del sondeo también podría verse afectado ya que los vecinos distantes son visibles sólo en los niveles de potencia más altos, en contraste con los vecinos más cercanos (que se ven con toda la gama de niveles de potencia). En consecuencia, tienen menos oportunidades de ser actualizados que sus vecinos

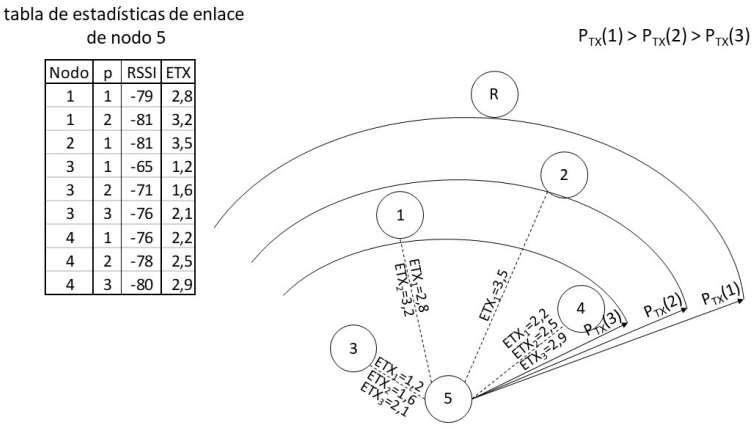


Figura 3.4 Distintos ETX y alcances para 3 niveles de potencia.

más cercanos[1] y por lo tanto, menos oportunidades de ser elegidos como padres preferidos (ya que sólo los vecinos actualizados son elegibles).

- La formación del DODAG se ralentiza. Los mensajes DIO se transmiten con un nuevo nivel de potencia cuando expira el temporizador Trickle-timer. Al aumentar el número de niveles, esto podría provocar un mayor retraso en la adhesión al DODAG y ralentizar su dinámica. Por ejemplo, un nodo con 5 niveles de potencia requeriría que expiren cinco temporizadores para anunciarse completamente a los demás.
- Se requiere más memoria. Algunas tablas (por ejemplo, estadísticas de enlaces) tendrán más filas, ya que están indexadas según la tupla dirección del nodo y nivel de potencia de transmisión. Esto implica una mayor huella de memoria.

En los siguientes apartados se explicarán los mecanismos propuestos para minimizar las posibles complicaciones.

3.2.1 Descripción básica del sistema propuesto

EL mecanismo entre capas propuesto opera en tres capas: red, enlace de datos y física, como se ilustra en la Figura 3.5.

A nivel físico, se asume que los nodos tienen un conjunto predefinido de P niveles de potencia de transmisión $i = 1,2,3,...,P$, donde 1 representa el nivel de potencia más alto y $P_{TX}(i)$ los vatios consumidos transmitiendo al nivel i. La potencia a la que transmitirá la capa física es fijada a través de los servicios de gestión, utilizando el punto de acceso al servicio (SAP) de la entidad de gestión de la capa física (PLME-SAP). La capa física

[1] El mecanismo de sondeo de Contiki selecciona un objetivo aleatorio para ser actualizado entre aquellos obsoletos cada vez que vence un temporizador.

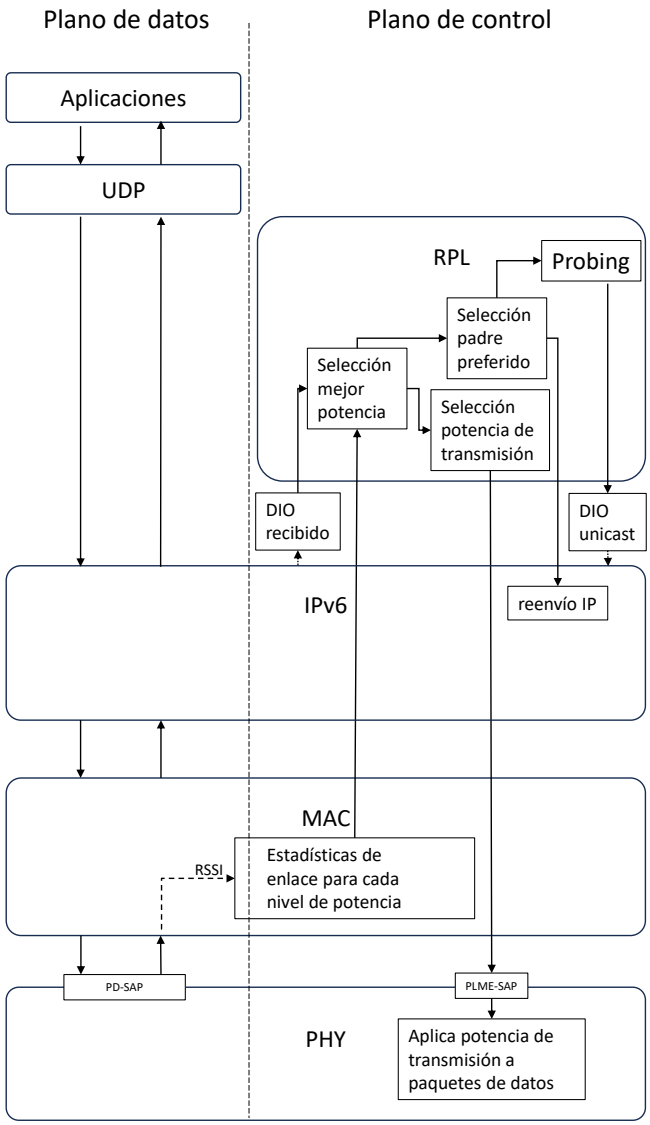


Figura 3.5 Esquema de capa cruzada en el dispositivo final.

también proporciona servicios de datos, a través del punto de acceso al servicio (SAP) de datos físicos (PD-SAP).

La subcapa MAC mantiene estadísticas de calidad del enlace para cada vecino descubierto (es decir, nueva dirección MAC). En el esquema propuesto, la subcapa MAC mantendrá P valores de ETX para cada vecino (es decir, $ETX(i), i = 1...P$) ya que los nodos tienen P niveles de potencia de transmisión disponibles.

En la capa de red, RPL se encarga de calcular la función objetivo para cada vecino n, recibiendo información de los mensajes DIO recibidos y de las estadísticas de enlace mantenidas por la subcapa MAC. La potencia a la que se transmitirán los paquetes de datos al nodo padre es decidida por RPL y comunicada a la capa física a través del punto de acceso al servicio (SAP) de la entidad de gestión de la capa física (PLME-SAP). Esta potencia es calculada a partir de la selección de la mejor potencia para cada nodo vecino n, que a su vez se obtiene a partir de los mensajes DIO recibidos de cada vecino n y de las estadísticas de enlace mantenidas por la subcapa MAC para cada vecino n y para cada nivel de potencia i. Estos cálculos también se utilizan en RPL para seleccionar al nodo padre como aquel nodo que minimiza la función objetivo.

Cuando se recibe un DIO de un vecino n, se le entrega a RPL, junto con el conjunto de valores $ETX(i), i = 1...P$ asociados con el enlace con ese vecino n. Tras ello, se calcula el nivel de potencia de transmisión óptimo asociado con el vecino emisor del DIO calculando el gasto de energía de transmisión más bajo esperado para el enlace con el nodo n (es decir, $\text{mín}_i\{ETX(i) \cdot P_{TX}(i)\}$, donde $P_{TX}(i)$ es la potencia de transmisión real para el nivel i). Se llamará a este valor el mejor nivel de potencia de transmisión p_n^* para el vecino n, y esa energía mínima para el enlace con el nodo n se usa como métrica de enlace local. Es decir, p_n^* para el vecino n es el valor de p que hace mínima la expresión (3.5), donde p son los niveles de potencia para los que se conoce el ETX con el vecino n.

$$p_n^* = \min_p \{ETX_n(p) \cdot P_{TX}(p)\} \tag{3.5}$$

A continuación, se procesa el DIO y se actualiza la función objetivo asociada al vecino n sumando el gasto de energía de transmisión más bajo esperado calculado anteriormente (es decir, métrica de enlace local) al valor recibido en el contenedor de métrica del mensaje DIO del vecino n (MC_n). Por lo tanto, el gasto de energía de transmisión más bajo esperado para llegar hasta el nodo raíz utilizando el vecino n, sería transmitiendo al nivel de potencia p_n^* y se corresponde con el de la expresión (3.6), donde el $ETX_n(i)$ es el ETX correspondiente al nivel de potencia i con el vecino n.

$$ETX_n(p_n^*) \cdot P_{TX}(p_n^*) + MC(n) \tag{3.6}$$

Finalmente, el vecino con el valor más bajo se elige como padre preferido. Una vez que se elige un padre preferido, la tabla de reenvío del nodo se actualiza en consecuencia y el algoritmo entre capas instruye a la capa física para que use el nivel p_n^* asociado con el padre preferido, para la transmisión de paquetes de datos de capa superior.

La Figura 3.6 muestra las capas correspondientes de la pila de protocolos del IoT y las tablas de soporte mantenidas por cada capa por el nodo 5 cuando sólo se consideran tres

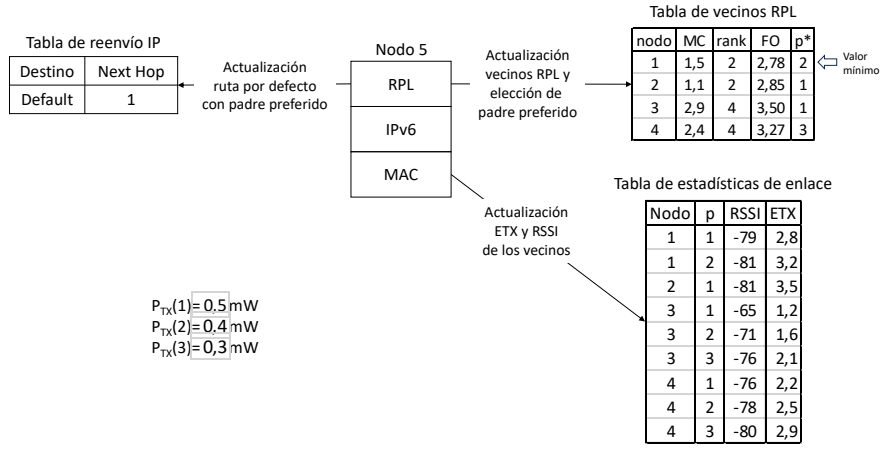


Figura 3.6 Ejemplo de tablas del nodo 5 con 3 niveles de potencia.

niveles de potencia de transmisión: alta (1), media (2) y baja (3). Se observa que el nodo 4 sería elegido como padre preferido y el p_n^* para ese nodo sería 3.

3.2.2 Mantenimiento de estadísticas del enlace para varios niveles de potencia

Esta sección amplía cómo se recopilan y mantienen las estadísticas de enlace para cada vecino y nivel de potencia de transmisión disponible. Por simplicidad, solo consideraremos tres niveles ($P = 3$) de potencia de transmisión: alta (1), intermedia (2) y baja (3). La extensión a más niveles es sencilla.

Mecanismos habilitadores del sistema propuesto

En el esquema propuesto en esta tesis, se requieren algunos cambios en el comportamiento de los siguientes elementos:

- Subcapa MAC de IEEE 802.15.4. Al recibir un paquete, la subcapa MAC debe conocer el nivel de potencia de transmisión utilizado por el remitente, para así poder actualizar las estadísticas correspondientes a ese nivel de potencia. Para esto, se define un *Elemento de Información* (*IE*) en el encabezado 802.15.4 para indicar el nivel de potencia utilizado en la transmisión de la trama (es decir, 1, 2 ó 3). Esta es una forma flexible de extender 802.15.4 de manera interoperable. Esta subcapa también debe poder indicar a la capa física que utilice una potencia de transmisión específica para paquetes de datos como resultado de una nueva actualización de padre preferido.
- Mensajes DIO de anuncio de RPL. Los nodos deben enviar los mensajes DIO a los distintos niveles de potencia de transmisión para que se puedan mantener estadísticas por nivel. Los anuncios de DIO de multicast se siguen enviando de acuerdo con

el temporizador Trickle-timer, pero en lugar de enviar un lote de mensajes DIO consecutivos en varios niveles de potencia, la potencia de transmisión se cambia cada vez en forma circular entre todos los niveles de potencia P. Esto disminuye el número de mensajes a costa de una reactividad más lenta. Sin embargo, en esta tesis se asume un escenario con nodos estáticos y se comienza con el nivel de potencia más alto $P_{TX}(1)$, por lo que el tiempo para unirse al DODAG no se ve comprometido.

- Sondeo RPL. Las estadísticas de ETX se actualizan periódicamente enviando un paquete DIO unicast a cada vecino obsoleto. En el esquema propuesto, un vecino se ve como una dirección y un nivel de potencia de transmisión específico. Así, el número de objetivos a sondear aumenta de N a $N \times P$, siendo N el número promedio de vecinos.
- Tabla de estadísticas de enlace: el ETX debe medirse para cada vecino y nivel de potencia de transmisión. Por lo tanto, la tabla de estadísticas de enlace ahora tendrá tantas entradas para cada vecino como niveles de potencia. Un ejemplo se ilustra en la Figura 3.6, donde el nodo 5 mantiene estadísticas de enlace con hasta 3 niveles de potencia con los vecinos 1, 2, 3 y 4.

Actualización de estadísticas

Cuando se recibe una nueva trama, la capa física genera una primitiva de indicación PD-DATA, que contiene el *RSSI* de la trama recibida y la envía a la subcapa MAC, que se encarga de mantener estadísticas de calidad del enlace.

Tras recibir el paquete, la subcapa MAC procesa el encabezado IEEE 802.15.4 y obtiene de ella tanto la dirección MAC origen como el nivel de potencia de transmisión utilizado para transmitir la trama (que se encuentra en la parte *IE* del encabezado). Luego, el $ETX_n(i)$ se inicializa (la primera vez que se descubre un vecino n y su nivel de potencia i) con el *RSSI* o se actualiza (para las entradas existentes) con cada paquete enviado a ese vecino en función de las tramas ACK (y un contador de retransmisión) [72]. Con estos valores, la subcapa MAC mantiene una tabla como la que se muestra en la Figura 3.6, donde la tupla *(dirección mac, nivel de potencia de transmisión)* se utilizará como índice en la tabla.

3.2.3 Especificación del algoritmo RPL de capa cruzada multinivel

El Algoritmo 1 detalla el comportamiento descrito en el apartado anterior, generalizándolo para el caso de P niveles de potencia de transmisión. Este algoritmo se ejecutará al recibir un DIO del nodo n.

El algoritmo tiene 4 partes principales. En la primera parte (líneas 1 a 8), se añaden entradas, si procede, a las tablas de estadísticas de enlace y a la tabla de vecinos RPL. En la segunda parte (líneas 9 a 10), se calcula el mejor nivel de potencia y la función objetivo asociado con el nodo n. En la tercera parte (líneas 11 a 14), se actualiza la entrada correspondiente al nodo n en la tabla de vecinos RPL, si se dan las condiciones necesarias. En la cuarta y última parte (líneas 15 a 29), si ha habido cambios (se ha actualizado la tabla de vecinos RPL), se busca el mejor vecino y potencia óptima, y si las estadísticas de nivel de enlace son frescas, se escoge como padre preferido, se actualiza la tabla de reenvío IP y se fija la potencia de transmisión. También se calculan los valores de $local\_rank$ (rango)

y *local_MC* (Metric Container) de los futuros mensajes DIO que el propio nodo que ha recibido el mensaje DIO tendrá que enviar a partir de este momento. Si las estadísticas de nivel de enlace no son frescas, se programa el envío de un sondeo urgente y si hay que cambiar el padre, se escoge el mejor padre y potencia óptima entre los vecinos que sí tienen las estadísticas frescas.

A continuación, se desarrolla el algoritmo con mayor nivel de detalle.

En la línea 1 se comprueba si ya existe una entrada en la tabla de estadísticas de enlace para el nodo n y el nivel de potencia P_{TX}, y si no existe, significa que es el primer DIO de ese nivel de potencia recibido de ese vecino, por lo que se calcula un valor inicial para el ETX a partir del RSSI (línea 2) y se añade una entrada a la tabla (línea 3). Si este vecino no se encuentra en la tabla de vecinos RPL (se comprueba en la línea 4), sería el caso de un vecino totalmente nuevo, y habría que añadir la entrada correspondiente a esa tabla (línea 5). Para esa entrada todavía no se ha calculado la función objetivo, y por eso se le pone un valor inicial de ∞, para ser sustituido por uno menor cuando se calcule. En la línea 6 se activa un flag para indicar que ha habido cambios en la tabla de vecinos RPL. El hecho de que la comprobación de la línea 4 esté dentro de la comprobación de la línea 1 se debe a que, si ya existía una entrada en la tabla de estadísticas de enlace, ya habría una entrada correspondiente a ese nodo en la tabla de vecinos RPL (y contendría la mejor potencia para ese vecino).

En la línea 9 se calcula p_n^* (es decir, el nivel que produce la métrica local más baja $ETX_n \cdot P_{TX}$) y se utiliza para calcular, en la línea 10, el nuevo valor FO_n de la función objetivo correspondiente al remitente del mensaje DIO (nodo vecino n), agregando la métrica local al valor recibido en la opción contenedor de métrica (MC Metric Container).

Estos valores calculados, p_n^*, FO_n, junto con los valores recibidos MC y rank, se actualizan en la entrada correspondiente al nodo n de la tabla de vecinos RPL en la línea 12, si el nuevo valor de la función objetivo es menor, o si el nivel que produce la métrica local más baja era el que ya existía para el nodo n (comprobación realizada en línea 11). En la línea 13 se activa un flag para indicar que ha habido cambios en la tabla de vecinos RPL.

Finalmente, si la tabla de vecinos RPL se ha actualizado (se comprueba el flag en la línea 15), se selecciona el mejor padre de entre todos los vecinos como aquel que minimiza la función objetivo (línea 16), y también se selecciona la mejor potencia para transmitir a ese nodo padre (línea 17). Lo anterior se mantiene si las estadísticas de enlace correspondientes a ese nodo y esa potencia son frescas (comprobación línea 18), en caso contrario se programa el envío de un sondeo urgente (mensaje DIO unicast) a ese nodo (línea 19) y si ese nuevo nodo no es el padre actual (se estaría escogiendo un nuevo padre con estadísticas no frescas) se busca el mejor padre y potencia óptima entre los vecinos que sí tienen las estadísticas frescas (líneas 21 y 22). Es decir, si el padre actual era el mejor pero no tiene las estadísticas frescas, se conserva ese padre (y se habrá programado un sondeo urgente para actualizarlas cuanto antes), pero no se cambia a un nuevo padre si sus estadísticas no están actualizadas. Las líneas 25 y 26 sirven para actualizar la tabla de reenvío IP con el padre escogido y fijar la potencia de transmisión en la capa física a la potencia óptima calculada para ese padre. En la línea 27 se calcula el *local_MC* (Metric Container) del propio nodo, como el valor de la función objetivo calculada para el nodo padre (que está almacenada en la tabla de vecinos RPL). En la línea 28 se calcula el

Algorithm 1: Algoritmo de capa cruzada con P niveles

Input : DIO,
rpl_neighbors_table
ip_forward_table,
link_stats_table,
$P_{TX}(p), \forall p \in [1:P]$

Initialize : n = source address from DIO,
P_{TX} = transmit power from DIO,
RSSI = rec. signal strength from DIO,
MC = metric container from DIO,
rank = rank from DIO,
changes = False

`/* add entries for DIO sender (n) */`

if ***not*** *link_stats_table.find(n, P_{TX})* **then**
  ETX = guess_from(RSSI);
  link_stats_table.add(n, P_{TX}, RSSI, ETX);
  if ***not*** *rpl_neighbors_table.find(n)* **then**
    rpl_neighbors_table.add(n, MC, rank, ∞, P_{TX});
    changes = True;
  end
end

`/* update best tx. power level for n and new objetive function */`

$p_n^* = \min_p \left(ETX_n(p) \cdot P_{TX}(p)\right), \forall p \in [1:P]$;
$FO_n = \mathrm{MC} + ETX_n(p_n^*) \cdot P_{TX}(p_n^*)$;

`/* update entry for DIO sender (n) */`

if $FO_n < rpl\_neighbors\_table.get\_FO(n)$ ***OR*** $p_n^* = rpl\_neighbors\_table.get\_p^*(n)$ **then**
  rpl_neighbors_table.update(n, MC, rank, FO_n, p_n^*);
  changes = True;
end

`/* update best parent */`

if *changes* **then**
  `/* select best from all neighbors */`
  best = $\min_m$(rpl_neighbors_table.get_FO(m)), $\forall m$;
  p_{best}^* = rpl_neighbors_table.get_p^*(*best*);
  if ***not*** *link_stats_table.is_fresh(best, p_{best}^*)* **then**
    schedule_urgent_probing(*best*, p_{best}^*);
    if *best* ***is not*** *current parent* **then**
      `/* select best from fresh neighbors */`
      best = $\min_m$(rpl_neighbors_table.get_FO(m)), $\forall$*fresh* m;
      p_{best}^* = rpl_neighbors_table.get_p^*(*best*);
    end
  end
  ip_forward_table.set_parent(*best*);
  set_local_tx_power(p_{best}^*);
  local_MC = rpl_neighbors_table.get_FO(best);
  local_rank = máx(rank + Δ, rpl_neighbors_table.get_FO(*best*));
end

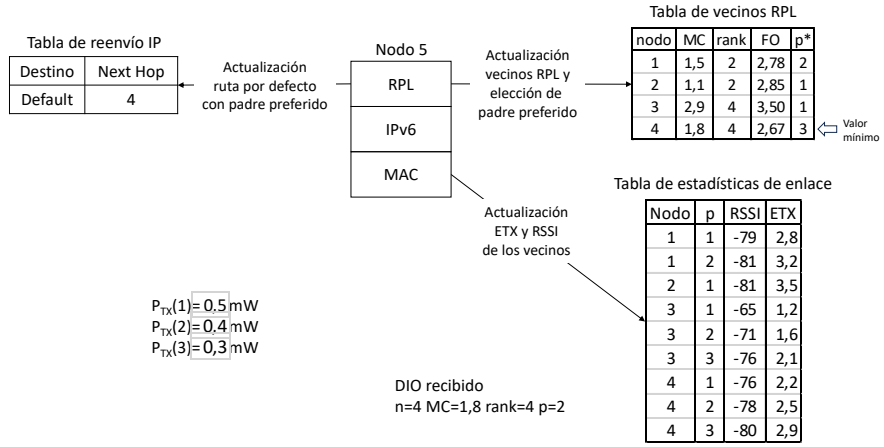


Figura 3.7 Ejemplo de tablas del nodo 5 tras recibir DIO de nodo 4.

local_rank (rango) del propio nodo, como el máximo entre el rango del nodo padre más un incremento mínimo (Δ) y el valor de la función objetivo calculado para ese padre. Estos valores calculados de *local_rank* (rango) y *local_MC* (Metric Container) son los que se usarán en los futuros mensajes DIO que el propio nodo, el que ha recibido el mensaje DIO, tendrá que enviar a partir de este momento.

Para facilitar la comprensión del algoritmo se va a evaluar con un ejemplo, suponiendo que el nodo 5 parte de la situación mostrada en la Figura 3.6 y que recibe un nuevo mensaje DIO del nodo 4 ($n = 4$) con $MC = 1{,}8$, $rank = 4$ y $p = 2$. El resultado se muestra en la Figura 3.7. Como ya hay una entrada en la tabla de estadísticas de enlace para ese nodo y nivel de potencia (la penúltima entrada) la condición de la línea 1 es falsa, y no se añade ninguna nueva entrada a las tablas. El cálculo de p_4^* de la línea 9 sigue siendo $p_4^* = 3$, ya que la tabla de estadísticas de enlace no ha cambiado y la mejor potencia para transmitir al nodo 4 es la de nivel 3, pero como ha cambiado el valor de MC recibido del nodo 4 ($MC = 1{,}8$) , ahora al calcular en la línea 10 la función objetivo a través del nodo 4 se obtiene $FO_4 = MC + ETX_4(p_4^*) \cdot P_{TX}(p_4^*) = 1{,}8 + 2{,}9 \cdot 0{,}3 = 2{,}67$. Al evaluar la condición de la línea 11, se cumple que el nuevo valor calculado de la función objetivo a través del nodo 4 $FO_4 = 2{,}67$ es menor que el antiguo $rpl\_neighbors\_table.get\_FO(4) = 3{,}27$, por lo que se actualiza la tabla de vecinos RPL en la línea 12 (nótese que $p_4^* = 3$ no cambia, ya que no ha cambiado la tabla de estadísticas de enlace). Si el nivel de potencia p_n^* calculado en la línea 9 fuera $p = 3$, entonces se actualizaría la tabla de vecinos RPL (en la línea 11 se cumple la segunda condición del OR) aunque el nuevo valor de la función objetivo fuera mayor, ya que los nuevos valores de MC y rank recibidos en el mensaje DIO del nodo 4 serían más actuales que los almacenados en la tabla. El resultado de esta actualización se puede observar en la tabla de vecinos RPL de la Figura 3.7. Como ha habido cambios en la tabla de vecinos RPL (se activó el flag en la línea 13), la condición de la línea 15 es cierta y

se calcula en la línea 16 el mejor de los vecinos como aquel que tiene un menor valor de la función objetivo en la tabla de vecinos RPL, en este caso $best = 4$, y en la línea 17 el mejor nivel de potencia para transmitir a ese nodo es $p^*_{best} = rpl\_neighbors\_table.get\_p^*(4) = 3$. Suponiendo que las estadísticas de enlace son frescas, es decir la entrada correspondiente al nodo 4 y nivel de potencia 3 está actualizada en la tabla de estadísticas de enlace (comprobación de la línea 18), entonces el nodo 4 sería el nuevo padre preferido. Si las estadísticas no fueran frescas, se programa un sondeo urgente al nodo 4 con nivel de potencia 3 (línea 19), y como el mejor vecino (nodo 4) no sería padre actual (se partía de la situación mostrada en la Figura 3.6 donde el padre actual es el nodo 1), tal como se comprobaría en la línea 20, entonces se buscaría el mejor vecino y nivel de potencia correspondiente (líneas 21 y 22) entre los nodos que tuvieran las estadísticas frescas. Finalmente, se toma el mejor vecino como padre preferido y se usa para actualizar, en la línea 25, la tabla de reenvío IP (se supone que era el nodo 4) e indicar al nivel físico, en la línea 26, el nivel de potencia al que hay que transmitir (nivel 3). También se calculan los valores que usará el propio nodo, el nodo 5, en los envíos de los futuros mensajes DIO. En la línea 27 se calcula el *local_MC* (Metric Container) del nodo 5 como $local\_MC =$ rpl_neighbors_table.get_FO$(4) = 2{,}67$. En la línea 28 se calcula el *local_rank* (rango) del nodo 5 como $local\_rank = \text{máx}\,(\text{rank} + \Delta, \text{rpl\_neighbors\_table.get\_FO}\,(4)) = \text{máx}\,((4+2),(2{,}67)) = 6$. Se ha supuesto que el incremento mínimo del rango es $\Delta = 2$.

3.3 Algoritmo de Probing propuesto para nodos multinivel

El valor del ETX con un vecino se va actualizando cuando se envían paquetes y se contabilizan los asentimientos recibidos o bien se inicializa a partir del RSSI, tal como se explicó en el apartado 3.2. Mediante el mecanismo de Probing, que es opcional en RPL, se persigue mantener actualizados los ETX con los distintos vecinos, almacenados en la tabla de estadísticas de enlace, aunque no se estén enviando paquetes de datos a esos vecinos. Si no se utilizara el mecanismo de Probing, sólo se tendría actualizado el ETX con el nodo padre, ya que en esta tesis sólo se contempla el caso de tráfico ascendente, y todos los paquetes se reenvían sólo a ese nodo.

No hay que confundir este mecanismo de Probing con otros mecanismos utilizados en RPL para el mantenimiento de rutas y adyacencias. Para el mantenimiento de rutas, los nodos transmiten mensajes DIO multicast de acuerdo con el algoritmo de goteo (temporizador Trickle [30]), que controla la velocidad de transmisión, ofreciendo un equilibrio entre reactividad a los cambios de topología y eficiencia energética. Para comprobar las adyacencias y detectar la ausencia de un vecino encontrado previamente, se pueden emplear dos mecanismos como la retroalimentación de eventos relacionados con la capa de enlace de datos (incluidos los asentimientos) y el mecanismo de detección de inalcanzabilidad de vecinos de IPv6 [82]. Este último permite a RPL rastrear activamente el "estado" de accesibilidad de sus vecinos, asegurando que pueda detectar cualquier falla mediante el envío de paquetes unicast. Este procedimiento externo se concentra en los enlaces más utilizados y minimiza la sobrecarga de mantener la adyacencia de encaminamiento. Pero esto también implica que no se verifica la adyacencia con padres no preferidos.

Por esta razón, algunas implementaciones de RPL (como la incluida en Contiki) permiten

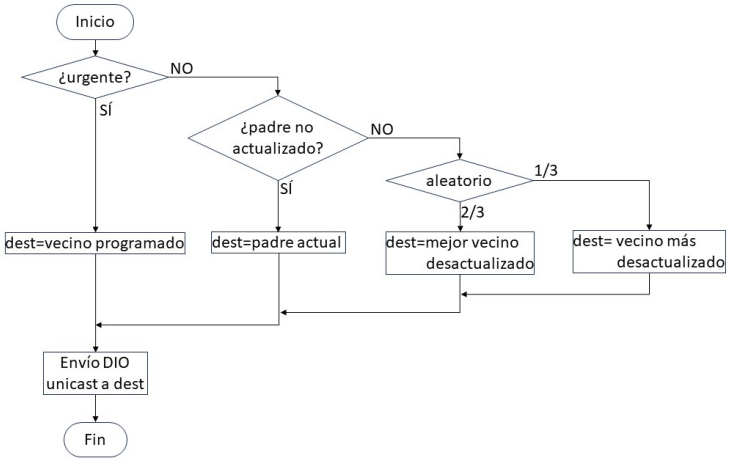


Figura 3.8 Sondeo de Probing original.

activar el mecanismo de Probing, que comprueba las adyacencias enviando periódicamente mensajes DIO unicast a los vecinos que tengan las estadísticas desactualizadas.

3.3.1 Algoritmo de Probing original

Una fila en la tabla de estadísticas de enlace correspondiente a un vecino se considera obsoleta o no fresca después de un cierto período de inactividad, en concreto, si ha pasado mucho tiempo desde que se recibió un asentimiento del vecino o se han enviado muchos paquetes a ese vecino sin recibir asentimiento. El algoritmo de sondeo predeterminado de Contiki tiene como objetivo mantener estadísticas actualizadas y al mismo tiempo evitar sobrecargar la red con paquetes de sondeo. Este algoritmo envía un paquete de prueba DIO unicast cada vez que expira un temporizador (aleatorio entre 45 y 135 s). Cuando expira el temporizador, el algoritmo elige un destino para actualizar de entre todas las entradas obsoletas en la tabla de estadísticas del nodo. El método de selección de destino es el siguiente. Lo primero es comprobar si hay un sondeo urgente programado. Ése será el caso en el que el mejor vecino no tenía la entrada actualizada y se programó ese sondeo urgente. Si no hay hay sondeo urgente, actualizar el padre actual tiene prioridad. Si el padre actual ya está actualizado, entonces con una probabilidad de 2/3, se elige la entrada (que esté desactualizada) con menor valor de la función objetivo y con una probabilidad de 1/3, se elige la entrada más desactualizada (es decir, la más antigua). Una vez seleccionado el destino, el algoritmo de sondeo envía un DIO de unicast a ese destino. Este método se puede describir mediante el diagrama de flujo de la Figura 3.8.

3.3.2 Limitaciones del algoritmo de Probing original con múltiples niveles de potencia

Cuando se trabaja con múltiples niveles de potencia, el número de entradas en la tabla de estadísticas de enlace aumenta, pasando de una entrada por vecino a múltiples entradas

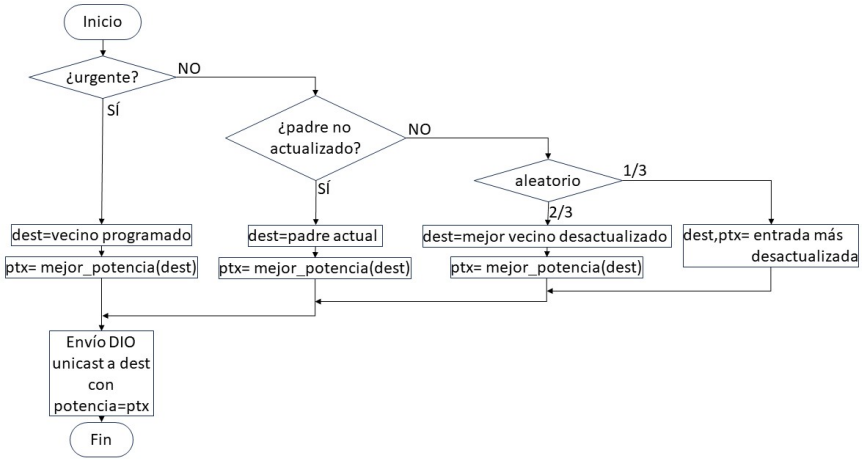


Figura 3.9 Sondeo de Probing con múltiples niveles de potencia.

por vecino, una por cada vecino y nivel de potencia. Por lo tanto, cuando hay que enviar un mensaje DIO unicast hay que seleccionar tanto el destino como la potencia a la que se va a transmitir, para actualizar la entrada correspondiente a esa pareja. La adaptación del algoritmo de Probing original a múltiples niveles de potencia se puede describir mediante el diagrama de flujo de la Figura 3.9. El uso del algoritmo de Probing original simplemente adaptado a múltiples niveles de potencia es problemático debido a la mayor cantidad de niveles de potencia disponibles, lo que se traduce en más entradas en la tabla de estadísticas de enlace. Si el máximo número de vecinos que se almacenan es N y el número de niveles de potencias con el que se trabaja es P, la tabla de estadísticas de enlace pasaría de tener N entradas a $N \cdot P$ entradas. Por ejemplo, si un nodo tiene cuatro vecinos y todos ellos están lo suficientemente cerca como para ser percibidos en cinco niveles de potencia, entonces su tabla de estadísticas de enlace tendría 20 entradas (frente a sólo 4 entradas cuando se usa un único nivel de potencia). Potencialmente, esto podría conducir a:

- Más tráfico de sondeo. Los paquetes de sondeo pueden aumentar en comparación con un único nivel de potencia ya que, con más entradas, es más probable que al expirar el temporizador el algoritmo de sondeo siempre encuentre una entrada desactualizada.

- Más tráfico a mayor potencia. Es más probable que los paquetes de sondeo se transmitan a mayor potencia, ya que las entradas con niveles de potencia más altos serán más abundantes y también tienden a tener un valor ETX más bajo que aquellos con niveles de potencia más bajos (ya que los vecinos distantes sólo son visibles a niveles de potencia más altos, mientras que los más cercanos son visibles en todos los niveles de potencia).

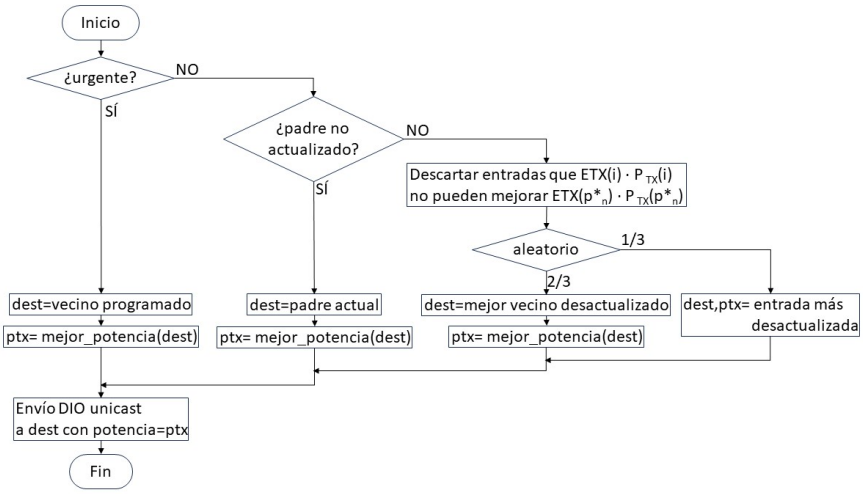


Figura 3.10 Sondeo de Probing Alternativo.

- Falta de equidad. Por el motivo anterior, la tabla de estadísticas de enlace tendrá más entradas para los nodos más cercanos que para los distantes. Como resultado, los nodos distantes tendrán menos oportunidades de actualizarse y ser elegidos como padres preferidos.
- Mayor consumo de memoria. El tener más entradas en la tabla de estadísticas de enlace implica que se consume más memoria, que es un recurso escaso en los dispositivos usados normalmente en las redes de sensores inalámbricas. La cantidad de memoria que se consume por cada entrada adicional en la tabla es de 2 octetos para RSSI, 2 octetos para ETX y 7 más para contadores y temporizadores, resultando un total de 11 octetos por entrada. Esta cantidad es relativamente pequeña comparada con una mota típica que puede tener 8 Kbytes de memoria RAM. No obstante, también se puede acotar el número máximo de vecinos mediante la constante NBR_TABLE_MAX_NEIGHBORS, para así reducir el tamaño de la tabla.

3.3.3 Especificación del algoritmo de Probing alternativo para nodos multinivel

Para minimizar los problemas anteriores, se propone un algoritmo de Probing Alternativo que reduce el conjunto de entradas candidatas elegibles para ser actualizadas, no teniendo en cuenta aquellas que, a pesar de estar desactualizadas, no mejorarían el valor de la función objetivo para un vecino. Esta modificación del algoritmo se puede describir mediante el diagrama de flujo de la Figura 3.10, y también se describe con mayor detalle en el Algoritmo 2.

El algoritmo tiene 2 partes principales, y termina devolviendo el nodo a sondear y la potencia a la que hay que transmitir la sonda (mensaje DIO unicast). En la primera parte (líneas 1 a 10) se comprueban los sondeos de alta prioridad, es decir, urgente y el padre actual. En la segunda parte (líneas 11 a 34) se revisa cada vecino de la tabla de vecinos RPL

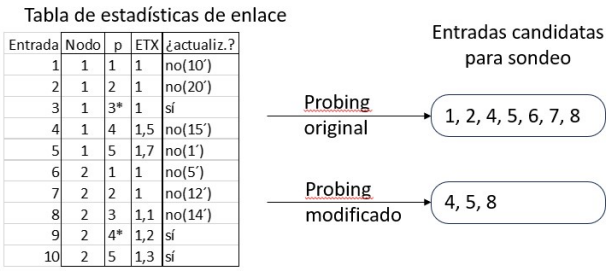
Algorithm 2: Modified get_probing_target pseudocode

Input : rpl_neighbors_table,
link_stats_table
Output target, p_{target}
:
;
// node for next probe and power level to use
1 **if** *any pending urgent probe* **then**
/* prioritize urgent probe requests */
2 *target* = pending urgent probe destination;
3 $p_{target} = p^*_{target}$;
4 **return** *target*, p_{target}
5 **end**
6 **if** *current parent is not fresh (parent,* p^*_{target} **then**
/* prioritize parent freshness */
7 *target* = *parent*;
8 $p_{target} = p^*_{parent}$;
9 **return** *target*, p_{target}
10 **end**
11 *rand_value* = *rand()*; // between 0 and 1
12 target = *init_value*;
13 **for** *each neighbor n in rpl_neighbors_table* **do**
/* exclude power levels higher than the optimal if ETX is already 1 */
14 **if** $ETX_n(p^*_n) > 1$ **then**
15 $p_{max} = \max(1, p^*_n - 1)$;
16 **else**
17 $p_{max} = p^*_n$;
18 **end**
19 **for** $p \in [p_{max} : P]$ **do**
20 **if** ***not*** *link_stats_table.is_fresh(n, p)* **then**
/* only non-fresh entries need probing */
21 **if** *rand_value* $\leq 2/3$ **then**
/* pick the best non-fresh with 2/3 probability */
22 **if** *target* = *init_value* ***OR*** (n,p) *has a better objetive function than* $(target, p_{target})$ **then**
23 target = n;
24 $p_{target} = p$;
25 **end**
26 **else**
/* pick the least recently updated non-fresh with 1/3 probability */
27 **if** *target* = *init_value* ***OR*** (n,p) *is more outdated than* $(target, p_{target}$ **then**
28 target = n;
29 $p_{target} = p$;
30 **end**
31 **end**
32 **end**
33 **end**
34 **end**

y procesa sólo las entradas de la tabla de estadísticas de enlace con los niveles de potencia que podrían mejorar la clasificación con ese vecino. El razonamiento que hay detrás del nuevo algoritmo es que una entrada candidata (vecino y nivel de potencia) con ETX = 1 no produce retransmisiones y no se puede mejorar (ETX = 1 es el valor mínimo), y por tanto sería inútil actualizar otras entradas para ese mismo vecino con un nivel de potencia mayor. Sin embargo, si para un vecino determinado, el valor óptimo tiene asociado un valor ETX superior a 1 (es decir, hay retransmisiones), el algoritmo incluye la entrada con el siguiente nivel de potencia superior para ese vecino como candidato a ser actualizado (con esa fila podría potencialmente mejorar la función objetivo), pero descarta las otras entradas de mayor potencia, porque es posible que con la entrada que se va a a actualizar se consiga llegar a ETX = 1.

A continuación, se desarrolla el algoritmo con mayor nivel de detalle.

En la línea 1 se comprueba si hay programado un sondeo urgente, que se habría programado en la línea 19 del Algoritmo 1, para el caso de que el mejor vecino no estuviera actualizado. En las líneas 2 y 3 se fijan el destino y el nivel de potencia, que se devuelven en la línea 4, terminando el algoritmo. Si no hay programado sondeo urgente, en la línea 6 se comprueba si el padre actual no está actualizado (es decir la entrada para la potencia óptima), y en ese caso las líneas 7, 8 y 9 son equivalentes a las 2, 3 y 4. Si no se da ninguno de los casos anteriores, en la línea 11 se asigna un valor aleatorio entre 0 y 1 a *rand_value*, que se usará en la línea 21 para realizar la acción correspondiente con una probabilidad de 2/3. En la línea 12 se asigna un valor inicial *init_value* a *target*, que es el valor que se devolvería si no se encontrara ningún nodo al que sondear. En la línea 13 comienza un bucle que recorre la tabla de vecinos RPL, es decir, una iteración para cada vecino n, aunque cada vecino puede tener varias entradas en la tabla de estadísticas de enlace. En las líneas 14 a 18 se calcula, para el nodo n, el máximo nivel de potencia p_{max} por encima del cual no se actualizarán las entradas correspondientes al nodo n en la tabla de estadísticas de enlace. Es decir, ese nivel p_{max} se usará para excluir de posibles sondeos, a las entradas cuyo nivel de potencia para el nodo n sea mayor a p_{max}. Para calcular p_{max}, se usa p_n^*, que es el nivel de potencia que minimiza la función objetivo para el nodo n (que está almacenado en la entrada para el nodo n en la tabla de vecinos RPL). Si el ETX correspondiente a esa potencia en la tabla de estadísticas de enlace, es decir $ETX_n(p_n^*)$, vale 1, se asigna esa potencia a p_{max} en la línea 17, ya que la condición de la la línea 14 es falsa. Si el ETX correspondiente es mayor que 1, entonces la condición de la línea 14 es cierta y se le asigna el siguiente nivel superior, es decir $p_n^* - 1$ en la línea 15. La razón de utilizar el operador máx con 1 es porque el máximo nivel de potencia es 1, y así se asegura que el nivel de potencia no toma el valor 0 (que no es válido). Una vez calculado p_{max}, con el bucle de la línea 19 se usa para recorrer las entradas del nodo n de la tabla de estadísticas de enlace que tengan una potencia inferior o igual al nivel p_{max}. Dentro de ese bucle sólo se analizan las entradas que no estén actualizadas, tal como se comprueba en la línea 20. La condición de la línea 21 se cumple con una probabilidad de 2/3, y en la línea 22 se compara la función objetivo del nodo n y potencia p con el nodo objetivo actual $target$ y potencia p_{target}, y si la función objetivo es menor se toma el nodo n como nuevo objetivo (línea 23), y como potencia a transmitir la potencia correspondiente a esa entrada p (línea 24). El hecho de compararlo también con *init_value* en la línea 22 es para encontrar el primer nodo que cumple las condiciones y sirva como caso inicial para el algoritmo. Si



Entrada	Nodo	p	ETX	¿actualiz.?
1	1	1	1	no(10′)
2	1	2	1	no(20′)
3	1	3*	1	sí
4	1	4	1,5	no(15′)
5	1	5	1,7	no(1′)
6	2	1	1	no(5′)
7	2	2	1	no(12′)
8	2	3	1,1	no(14′)
9	2	4*	1,2	sí
10	2	5	1,3	sí

Figura 3.11 Ejemplo de Probing Alternativo.

no se cumple la condición del la línea 21, con una probabilidad de 1/3 se pasa a la línea 27 y si el nodo n se actualizó con el nivel de potencia p por última vez antes que el nodo objetivo actual $target$ con p_{target}, se toma el nodo y la potencia en las líneas 28 y 29, de forma similar a las líneas 23 y 24. Esta información de la última vez que se actualizó un nodo a una determinada potencia está almacenada en la tabla de estadísticas de enlace. La comparación con $init\_value$ en la línea 27 es similar a la de la línea 22. Finalmente, en la línea 35, se devuelve el nodo al que hay que enviar el sondeo (DIO unicast) y la potencia a la que hay que transmitir. El mensaje de sondeo servirá para actualizar una entrada no actualizada de la tabla de estadísticas de enlace. El algoritmo devuelve $init\_value$ si no hubiera entradas no actualizadas, o las que hubiera tuvieran una potencia descartada, y en ese caso no se realizaría ningún sondeo.

Para ilustrarlo, se va a usar como ejemplo la tabla de estadísticas de enlace de un nodo con dos vecinos (nodos 1 y 2) y cinco niveles de potencia (donde 1 es el nivel de potencia más alto). Esto se muestra en la Figura 3.11, donde la tercera entrada (nodo 1 y nivel de potencia 3) es el padre actual y las entradas marcadas con * son el mejor nivel de potencia p_n^* para ese vecino n, es decir, las que producen menor $ETX(p_n^*)\dot{P}_{TX}(p_n^*)$. Luego, suponiendo que las entradas 3, 9 y 10 estén actualizadas, el algoritmo de Probing original tendría que elegir un objetivo entre 7 candidatos, mientras que el algoritmo de Probing Alternativo tendría que elegir un objetivo entre 3 candidatos, ya que se descartan las entradas 1, 2, 6 y 7. Se descartan las entradas 1 y 2 porque para el nodo 1, el ETX correspondiente a la potencia óptima vale 1, y por lo tanto las entradas 1 y 2 no pueden mejorar el producto $ETX \cdot P_{TX}$, ya que la potencia aumentaría y ETX no puede bajar de 1. Se descartan las entradas 6 y 7 porque para el nodo 2, el ETX correspondiente a la potencia óptima es mayor que 1, y por lo tanto se toma como candidata la entrada con el siguiente nivel de potencia (entrada 8), pero no las entradas con niveles superiores (entradas 6 y 7). El método de selección de destino, que es el mismo en ambos esquemas, seleccionaría con una probabilidad de 1/3 el más antiguo (entrada 2 en el algoritmo de Probing original y entrada 4 en el algoritmo de Probing Alternativo) o, con una probabilidad de 2/3, la fila con el valor de la función objetivo más bajo (que dependería de los valores de MC_n y $ETX_n(p) \cdot P_{TX}(p)$).

Para comprobar que la ejecución del algoritmo coincide con lo explicado, se va a ejecutar paso a paso con el ejemplo de la Figura 3.11. Se supone que no hay programado un sondeo

urgente y la condición de la línea 1 es falsa. El padre actual es el nodo 1, con nivel de potencia 3, y al estar actualizado, la condición de la línea 6 es falsa. En la línea 11 se toma un valor aleatorio, por ejemplo $rand\_value = 0{,}7$, por lo que en este caso se buscará la entrada menos actualizada . En la linea 12 se asigna el valor inicial $init\_value$ a $target$, y se pasa a los bucles anidados de las líneas 13 y 19. El primer bucle, el de la línea 13, recorre los dos vecinos, es decir $n = 1$ y $n = 2$.

- En la primera iteración, con $n = 1$, el valor de p_{max} se calcula en la línea 17 como p_1^* que vale 3. Esto es así porque la condición de la línea 14 es falsa, ya que $ETX_n(p_n^*) = ETX_1(p_1^*) = ETX_1(3) = 1$. El segundo bucle, el de la línea 19, recorre el intervalo $[p_{max} : P] = [3{,}5]$, es decir,las 3 potencias más bajas del nodo 1: $p = 3$, $p = 4$ y $p = 5$ (por lo que se descartan las entradas 1 y 2, que se usaban en el algoritmo original).
  - En la primera iteración $(n = 1, p = 3)$, la condición de la línea 20 es falsa porque la entrada correspondiente (entrada 3) está actualizada, y se pasa a la siguiente iteración.
  - En la segunda iteración del bucle de la línea 19 $(n = 1, p = 4)$, la condición de la línea 20 es cierta porque la entrada 4 no está actualizada, y se pasa a la línea 21. La condición de la línea 21 en este caso es falsa (ya que $rand\_value = 0{,}7 > 2/3$), y se pasa a la línea 27, cuya condición es cierta (ya que $target = init_{value}$), tomándose los valores $(n = 1, p = 4)$ para $(target, p_{target})$ (en las líneas 28 y 29).
  - En la tercera iteración del bucle de la línea 19 $(n = 1, p = 5)$, al igual que en la iteración anterior, la condición de la línea 20 es cierta (la entrada 5 no está actualizada) y la de la 21 es falsa (ya que $rand\_value = 0{,}7 > 2/3$). Se pasa a la línea 27, donde se comprueba si $n = 1$,$p = 5$ está más desactualizada que $(target = 1, p_{target} = 4)$, y al no estarlo, no se ejecutan las líneas 28 y 29, y se conservan los valores $(target = 1, p_{target} = 4)$.
- En la segunda iteración del bucle de la línea 13, con $n = 2$, el valor de p_{max} se calcula en la línea 17 como $p_2^* - 1$ que vale 3. Esto es así porque la condición de la línea 14 es cierta, ya que $ETX_n(p_n^*) = ETX_2(p_2^*) = ETX_2(4) = 1{,}2 > 1$. El segundo bucle, el de la línea 19, recorre el intervalo $[p_{max} : P] = [3{,}5]$, es decir, las 3 potencias más bajas del nodo 3: $p = 3$, $p = 4$ y $p = 5$ (por lo que se descartan las entradas 6 y 7, que se usaban en el algoritmo original).
  - En la primera iteración $(n = 2, p = 3)$, la condición de la línea 20 es cierta porque la entrada correspondiente (entrada 8) está desactualizada, y se pasa a la línea 21. La condición de la línea 21 en este caso es falsa (ya que $rand\_value = 0{,}7 > 2/3$), y se pasa a la línea 27, donde se comprueba si $n = 2$,$p = 3$ está más desactualizada que $(target = 1, p_{target} = 4)$, y al no estarlo, no se ejecutan las líneas 28 y 29, y se conservan los valores $(target = 1, p_{target} = 4)$.
  - En la segunda iteración del bucle de la línea 19 $(n = 2, p = 4)$, la condición de la línea 20 es falsa porque la entrada correspondiente (entrada 9) está actualizada, y se pasa a la siguiente iteración.

– En la tercera iteración del bucle de la línea 19 ($n = 2, p = 5$), al igual que en la iteración anterior, la condición de la línea 20 es falsa porque la entrada correspondiente (entrada 10) está actualizada, y se sale de los dos bucles, pasando a la línea 35, donde el algoritmo termina y se devuelven los últimos valores calculados ($target = 1, p_{target} = 4$), es decir, los correspondientes a la entrada 4.

Una vez terminado el algoritmo con el valor aleatorio de la línea 11 por encima de 2/3 (probabilidad de 1/3), se comprueba que el valor devuelto se corresponde con la entrada 4, que es la más desactualizada de las entradas candidatas del ejemplo de la Figura 3.11, (la entrada 4 es la más desactualizada entre las entradas 4, 5 y 8).

Si al ejecutar el algoritmo, el valor aleatorio de la línea 11 estuviera por debajo de 2/3 (probabilidad de 2/3), entonces devolvería la entrada con mejor valor de la función objetivo de las entradas candidatas.

Tal como se ha especificado, el esquema propuesto de Probing Alternativo, reduce significativamente la cantidad de destinos que se actualizarán en comparación con el algoritmo de Probing original, lo que mejora los aspectos negativos antes mencionados (es decir, más tráfico, mayor potencia en DIO y falta de equidad), aunque no soluciona el incremento de uso de memoria.

3.4 Conclusiones del capítulo

En este capítulo, en el apartado 3.1, se han descrito inicialmente las limitaciones del funcionamiento básico de RPL, cuando los nodos transmiten a una potencia única, y en ese caso, la minimización de $\sum ETX_j$ es equivalente a la minimización de la energía, tal como se vio en la fórmula (3.4). Pero cuando se quiere aprovechar la capacidad que tienen algunos nodos de poder transmitir a múltiples potencias, y así ahorrar más energía transmitiendo a la potencia mínima necesaria, se encuentra que la potencia de transmisión afecta al conjunto de vecinos accesibles, y a los ETX.

El esquema propuesto en el apartado 3.2 tiene en cuenta esa particularidad y minimiza el consumo de energía minimizando $\sum ETX_j(i) \cdot P_{TX}(i)$, tal como se muestra en el Algoritmo 1 de capa cruzada, que refleja el funcionamiento expuesto en la Figura 3.5. En esta figura se observa que en el módulo RPL, mediante los mensajes DIO recibidos y las estadísticas de enlace, se selecciona el padre preferido y se fija la potencia a la que la capa física debe transmitir los paquetes de datos. En este esquema, las estadísticas de enlace han tenido que ser ampliadas para tener en cuenta el ETX para cada potencia y para cada vecino, y estos ETX, junto con la potencia correspondiente, se han usado en el cálculo de la función objetivo.

También en el módulo RPL se eligen los nodos a los que enviar mensajes DIO unicast para el sondeo (Probing), destinados a mantener actualizadas las estadísticas de enlace, y esta elección presenta limitaciones con múltiples niveles de potencia que afectan al rendimiento. En la sección 3.3 se presenta el Algoritmo 2 de Probing alternativo, que elimina la mayoría de esas limitaciones.

Con estos dos algoritmos propuestos, el sistema previsiblemente reducirá el consumo de energía existente, si bien es cierto que la introducción de este nuevo código también añade

complejidad al sistema. Por lo tanto se hace necesario realizar una implementación del esquema propuesto y validar experimentalmente, mediante una batería de pruebas, el ahorro de energía. El siguiente capítulo está dedicado a describir cómo es la implementación que se ha usado para la batería de pruebas.

4 Implementación de la propuesta en Contiki-NG

Para validar la propuesta anterior y cuantificar sus beneficios, se necesita realizar experimentos. La implementación usada en esta tesis está basada en Contiki-NG [36], que comenzó como una reescritura importante de la versión 2017 de Contiki [37], centrándose en las funcionalidades más importantes y estables. De forma predeterminada, Contiki-NG implementa una versión ligera de RPL llamada RPL-lite [83]. Tal como se indicó en el apartado 2.2.6, esta versión tiene limitaciones, pero éstas no afectan al estudio del tráfico ascendente que se hace en esta tesis.

4.1 Contiki-NG

Contiki-NG tiene su propia estructura para organizar los protocolos de comunicación vistos en el apartado 2.1, en concreto en la Figura 2.1. Esta estructura se puede ver en la Figura 4.1, donde por un lado se tiene cómo se organizan los diversos protocolos en las capas de comunicaciones de Contiki-NG, y por otro lado, asociado a cada capa, están los módulos que implementan los protocolos que se pueden configurar en Contiki-NG. Se puede apreciar que algunas de las capas permiten elegir qué módulo utilizar.

4.2 Arquitectura de red de Contiki-NG

Una vez se ha visto la estructura de las capas de comunicaciones de Contiki-NG, ahora se va a dar una visión general de cómo funcionan todas estas capas con sus módulos y cómo interaccionan entre sí para hacer posible el sistema de comunicaciones en Contiki-NG. En la Figura 4.2 se muestran los módulos configurados y sus interacciones. Cada capa de comunicaciones está configurada con los módulos utilizados en esta tesis, que se detallan en los apartados siguientes.

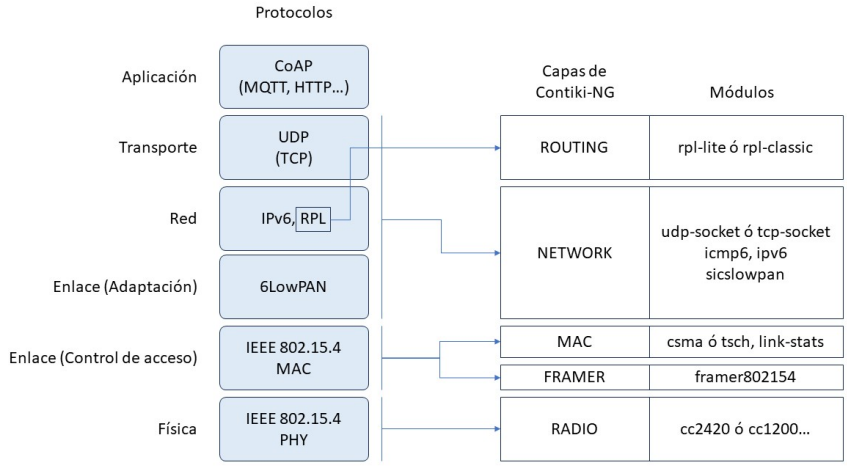


Figura 4.1 Capas de comunicaciones de Contiki-NG y sus módulos.

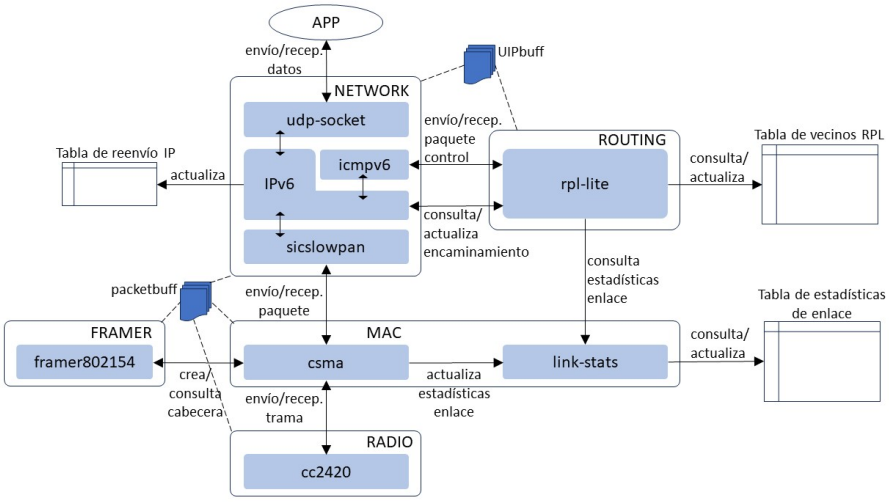


Figura 4.2 Componentes principales de la pila de comunicaciones de Contiki-NG.

4.2.1 Capa NETWORK

Esta capa es la que se encarga de la comunicación a nivel de transporte y de red, dejando para otra capa el encaminamiento. Esta capa es la encargada de crear y recibir los paquetes de red que contienen los datos enviados y recibidos por la capa de aplicación. Para la creación y lectura de los paquetes de red, todos los módulos de esta capa utilizan el búfer

UIPbuff, donde cada módulo se encarga de montar o leer su parte de la cabecera del paquete. Este búfer es compartido con el módulo rpl-lite de la capa ROUTING. Los módulos de los que se compone esta capa son:

- **udp-socket**. Este módulo permite la comunicación mediante UDP entre motas, para el envío y recepción de los paquetes de la aplicación. Este módulo usa los servicios ofrecidos por el módulo ipv6.
- **icmp6**. Se encarga de enviar y recibir los mensajes de control de IPv6 (ICMv6). En el trabajo desarrollado en esta tesis sólo se envían los mensajes DIO, DIS y DAO de RPL, que son mensajes ICMPv6.
- **ipv6**. Este es el núcleo de la capa NETWORK ya que es el módulo principal, responsable de la creación y procesamiento de datagramas y la función de reenvío. También mantiene una lista con las direcciones de red y enlace de cada vecino (tabla de vecinos IPv6) y la tabla de reenvío (asistido por la capa de ROUTING).
- **sicslowpan**. Realiza la compresión/descompresión de encabezados de los paquetes IPv6, y también la fragmentación/reensamblaje si es necesario. Una vez que un paquete está listo, lo lee del búfer UIPbuff y lo coloca como carga útil en un búfer llamado packetbuff, accesible para la subcapa MAC.

4.2.2 Capa ROUTING

Este capa se ocupa del encaminamiento, utilizando el módulo RPL-Lite.

- **rpl-lite**. Genera y procesa mensajes de control RPL (DIO, DIS y DAO) que se pasan hacia/desde el módulo icmp6 a través del búfer UIPbuff. Cuando se selecciona un padre preferido, este módulo le indica al módulo ipv6 que actualice su tabla de reenvío. También mantiene la tabla de vecinos RPL con información relevante de cada vecino (por ejemplo, rango, contenedor de métrica MC, etc.) que se identifican por su dirección MAC. Para estadísticas locales como ETX, el módulo RPL-lite consulta el módulo de estadísticas de enlace link-stats.

4.2.3 Capa MAC y FRAMER

Estas capas contienen una implementación básica de IEEE 802.15.4 con CSMA/CA, y sus módulos son:

- **csma**. Lee paquetes colocados por sixclowpan en el packetbuff y compone tramas IEEE 802.15.4 utilizando el módulo framer. Luego, envía la trama a la capa de radio (utilizando el búfer de paquetes compartido) de acuerdo con el algoritmo CSMA/CA, esperando la recepción de ACK y devolviendo el estado de la transmisión. Cada vez que se envía o recibe una trama, llama al módulo de estadísticas de enlace link-stats para actualizar RSSI y/o ETX.
- **framer**. Es el módulo que se encarga de gestionar las cabeceras IEEE 802.15.4. Cuando se va a enviar una trama, añade la cabecera al búfer packetbuff, basándose en la configuración de la capa MAC. Cuando se recibe una trama, la analiza, extrayendo

la cabecera y almacenando la trama en el búfer, hace las comprobaciones necesarias, y si todo es correcto informa a la capa superior y al módulo link-stats.

- **link-stats**. Este módulo calcula y mantiene estadísticas de enlace utilizando información de la capa MAC con cada paquete enviado (número de retransmisiones, el estado de la transmisión, etc.) o recibido (RSSI). Almacena estadísticas para cada dirección MAC descubierta, en la tabla de estadísticas de enlace, que incluye el valor de RSSI, ETX e información adicional, como contadores del número de transmisiones, asentimientos o un indicador de actualización. Para cada vecino esta información se almacena en una estructura del tipo `struct link_stats` que se muestra en el Código 4.1.

```
struct link_stats {
  clock_time_t last_tx_time ; /* Last Tx timestamp */
  uint16_t etx ;              /* ETX using ETX_DIVISOR as fixed point divisor */
  int16_t rssi ;              /* RSSI ( received signal strength ) */
  uint8_t freshness ;         /* Freshness of the statistics */
};
```

Código 4.1 Fragmento de link-stats.h original.

El valor de ETX se inicializa (la primera vez que se descubre un vecino) o se actualiza (para las entradas existentes) con cada paquete intercambiado con ese vecino de la siguiente manera:

– Inicialización de ETX. Por defecto, ETX se inicia con un valor predeterminado (por ejemplo, 2) o se puede configurar para que el valor inicial se calcule a partir del RSSI del paquete recibido, en cuyo caso $ETX = 1$ si $RSSI \geq -60dBm$, $ETX = 3$ si $RSSI \leq -80dBm$, y un valor intermedio si $-80 \leq RSSI \leq -60$ según la siguiente fórmula:

$$ETX = \frac{30}{RSSI + 90} \tag{4.1}$$

En esta tesis se ha usado la inicialización a partir del RSSI, que refleja mejor las características del enlace, en vez de asignarle un valor fijo.

– Actualización ETX. Se puede configurar para que se realice de dos formas posibles: utilizando la Media Móvil Ponderada Exponencial (EWMA) (que es la predeterminada) o utilizando los contadores de transmisiones y asentimientos. En ambos métodos, un paquete no asentido añade una penalización de 12 al contador del número de transmisiones. En el método EWMA, el ETX para la i-ésima actualización se calcula como:

$$ETX(i) = ETX(i-1) \cdot (100 - \alpha) + numtx \cdot \alpha \tag{4.2}$$

donde $\alpha = 10$ si las estadísticas del enlace están actualizadas o $\alpha = 25$ en caso contrario (son los valores predeterminados), y $numtx$ es el contador de

PA_LEVEL	TXCTRL register	Output Power [dBm]	Current Consumption [mA]
31	0xA0FF	0	17.4
27	0xA0FB	-1	16.5
23	0xA0F7	-3	15.2
19	0xA0F3	-5	13.9
15	0xA0EF	-7	12.5
11	0xA0EB	-10	11.2
7	0xA0E7	-15	9.9
3	0xA0E3	-25	8.5

Figura 4.3 Niveles de potencia para el transceptor radio CC2420.

número de transmisiones de un paquete (hasta que se recibe el asentimiento). En esta tesis se ha usado el método EWMA, que permite ahorrar memoria ya que no hay que almacenar los contadores de paquetes correspondientes para cada vecino y nivel de potencia.

4.2.4 Capa RADIO

El transceptor de radio utilizado en esta tesis es el CC2420, y está implementado en el simulador. Envía y recibe tramas hacia/desde packetbuff y proporciona estadísticas de la capa física, para ser servidas a la capa mac. Una de las características más interesantes que presenta este transceptor es la posibilidad de transmisión a diferentes niveles de potencia. Esta característica es utilizada en el desarrollo de la tesis para transmitir a potencias inferiores a la máxima, buscando el ahorro de energía. Los paquetes se transmiten utilizando el nivel de potencia indicado en un registro, tal como se muestra en la Figura 4.3, obtenida de las especificaciones del fabricante [84].

4.3 Modificaciones al código original de Contiki-NG

Esta sección describe las principales modificaciones realizadas en la pila de red original.

4.3.1 Mecanismo para modificar el nivel de potencia de transmisión

Se ha agregado una variable global que permite al módulo rpl-lite indicar el nivel de potencia de transmisión que deberá utilizar la radio. La capa MAC usará esta variable para modificar el registro que la capa RADIO utiliza para decidir el nivel de potencia de transmisión. La capa MAC modifica el registro justo antes de indicarle a la capa RADIO que transmita una nueva trama (que contendrá los datos de la aplicación), y lo restablece a su valor predeterminado después de la transmisión (máxima potencia). Esto se hace así para usar el asentimiento de tramas MAC por hardware que tienen implementado por defecto algunos chips radio como el CC2420, y que los asentimientos se transmitan siempre a máxima potencia. Si no fuera así, puede ocurrir que un nodo transmita una trama a una potencia mayor que la de su receptor, y el asentimiento del receptor nunca llegue al nodo transmisor de la trama, debido a su baja potencia de transmisión. La desactivación de los asentimientos por hardware provoca retrasos debido al procesamiento del software

y a la sobrecarga en condiciones de tráfico intenso, por lo que se ha decidido habilitarlos, manteniendo la radio siempre a máxima potencia, excepto cuando sea necesario.

4.3.2 Cambios en IEEE802.15.4

Además de poder transmitir tramas a una potencia específica, también es necesario que el receptor de una trama sepa a qué potencia fue transmitida, para poder relacionar las características de esa trama con las estadísticas asociadas al nodo transmisor y la potencia a la que se transmitió. Las tramas que necesitan informar sobre la potencia a la que fueron transmitidas incluyen un nuevo encabezado de elemento de información (IE) (5 bytes), que extiende el del IEEE 802.15.4, para indicar el nivel de potencia de transmisión utilizado. Además de la trama, la estructura de datos del búfer packetbuff almacena información adicional como RSSI (de una trama recibida) o campos de encabezado como el número de secuencia, etc. Se aprovecha este búfer para intercambiar información entre capas, relacionada con la potencia, introduciendo dos nuevas variables. Una de las variables indica si se va a transmitir a un nivel de potencia distinto de la potencia máxima, y la otra el nivel de potencia que se utilizará para transmitir la trama. Estas variables se examinan para manejar el contenido del encabezado de IE antes de la transmisión de una trama. De forma similar, después de la recepción de una trama, estas variables indican el nivel de potencia de la trama recibida.

4.3.3 Modificaciones en el cálculo de ETX

El ETX debe iniciarse y actualizarse para cada nodo vecino y nivel de potencia de transmisión, es decir para cada tupla [dirección mac, nivel de potencia]. Se ha modificado la estructura de datos predeterminada para tener en cuenta los distintos niveles de potencia. Ahora, para cada vecino, en vez de almacenar una estructura tal como se vio en el Código 4.1, se almacena una tabla de estructuras, de nombre `ptx_stats`, donde cada elemento es del tipo `struct link_stats_ptx`, tal como se muestra en el Código 4.2, que es parte del fichero `os/net/link-stats.h`. El tamaño de la tabla es el número de niveles de potencia que se usan, dado por la constante `NUM_PTX_LVLS`.

```
struct link_stats_for_ptx {
  clock_time_t last_tx_time ; /* Last Tx timestamp */
  uint16_t etx ;              /* ETX using ETX_DIVISOR as fixed point divisor */
  int16_t rssi ;              /* RSSI ( received signal strength ) */
  uint8_t freshness ;         /* Freshness of the statistics */
};

struct link_stats {
  struct link_stats_for_ptx ptx_stats [NUM_PTX_LVLS];
};
```

Código 4.2 Fragmento de link-stats.h para varios niveles de potencia.

Otras modificaciones en el módulo ETX son:

- antes de calcular el ETX para un paquete saliente o actualizar el RSSI para un paquete entrante, se obtiene el nivel de potencia de transmisión de dicho paquete, para poder actualizar la entrada correcta [dirección mac, nivel de potencia] en la tabla.

  Para un paquete saliente, el ETX se calcula en la función link_stats_packet_sent , parte de la cual (sobre todo lo relacionado con varios niveles de potencia) se muestra en el Código 4.3, y sus tres parámetros se pueden observar en la línea 2: la dirección MAC destino (MAC del vecino) lladdr , el estado de la transmisión del paquete status y el número de veces numtx que se ha transmitido ese paquete hasta recibir el asentimiento. La potencia a la que se ha enviado un paquete está almacenada en la cabecera del mismo, y se obtiene de packetbuff mediante la función packetbuff_attr en la línea 10. La potencia se convierte al índice correspondiente en la línea 10 mediante la función get_ptx_lvls_index y se almacena en la variable index , que se utiliza para acceder al elemento correspondiente de la tabla nbr_stats −>ptx_stats[] en la línea 11. La dirección de ese elemento se almacena en el puntero stats en la línea 11, y a partir de él ya se puede acceder a los distintos valores correspondientes a la entrada [dirección mac, nivel de potencia]=[lladdr , index]. El valor del puntero nbr_stats se obtiene mediante la función nbr_table_get_from_lladdr en la línea 8, que busca en la tabla global link_stats la dirección MAC destino y devuelve la dirección de una estructura cuyo campo ptx_stats es la tabla de NUM_PTX_LVLS elementos con las estadísticas correspondientes a cada nivel de potencia vista en el Código 4.2. Antes de actualizar el ETX en stats −>etx, se comprueba si vale 0 (valor que se utiliza en el código para indicar que el ETX no está inicializado) en la línea 13, y en ese caso hay que inicializarlo a partir del RSSI en la línea 14, usando la función guess_etx_from_rssi que toma el valor de RSSI de stats y aplica la fórmula 4.1 (aunque en el fichero está la posible inicialización con un valor por defecto, por claridad no se muestra aquí, ya que ese método de inicialización no se usa en esta tesis). Si el estado de transmisión del paquete status indica que no se ha recibido asentimiento (MAC_TX_NOACK) en la línea 18, entonces se incrementa en 12 (ETX_NOACK_PENALTY) el número de transmisiones en la línea 19, tal como se explicó en el apartado 4.2.3. En la línea 22, para la actualización del ETX según la fórmula 4.2, se usa la variable packet_etx como $ETX(i-1)$ con el valor del número de transmisiones numtx. El hecho de multiplicarlo por 128 (ETX_DIVISOR) es porque en los mensajes DIO el valor del ETX se codifica con desplazamiento de 7 bits. En la línea 24 se usa la variable ewma_alpha como α de la fórmula 4.2, y su valor depende de si las estadísticas correspondientes a la tupla [lladdr , index], que están apuntadas por stats , están actualizadas, en cuyo caso toma el valor 10 (EWMA_ALPHA), y el valor 25 (EWMA_BOOTSTRAP_ALPHA). Finalmente, en la línea 25 se actualiza el ETX, aplicando la fórmula 4.2 y modificando stats −>etx.

```
1  void
2   link_stats_packet_sent (const linkaddr_t *lladdr, int status, int numtx)
3  {
4
5  struct link_stats *nbr_stats;
6  struct link_stats_for_ptx *stats;
```

```
7
8   nbr_stats = nbr_table_get_from_lladdr(link_stats, lladdr);
9
10  int index = get_ptx_lvls_index(packetbuf_attr(PACKETBUF_ATTR_PTX));
11  stats = &nbr_stats->ptx_stats[index];
12
13  if(stats->etx == 0){
14    stats->etx = guess_etx_from_rssi(stats);
15  }
16
17  /* Add penalty in case of no-ACK */
18  if(status == MAC_TX_NOACK) {
19      numtx += ETX_NOACK_PENALTY;
20  }
21
22  uint16_t packet_etx = numtx * ETX_DIVISOR;
23  /* ETX alpha used for this update */
24  uint8_t ewma_alpha = ptx_link_stats_is_fresh(stats) ? EWMA_ALPHA :
        EWMA_BOOTSTRAP_ALPHA;
25  stats->etx = ((uint32_t)stats->etx * (EWMA_SCALE - ewma_alpha) +
26      (uint32_t)packet_etx * ewma_alpha) / EWMA_SCALE;
```

Código 4.3 Fragmento de link-stats.c: cálculo de ETX para varios niveles de potencia.

- la frescura y el instante de la última transmisión ahora también se almacena para cada nivel de potencia de cada nodo, es decir, para cada tupla [dirección mac, nivel de potencia]. Por ejemplo, en el Código 4.3 se puede observar que la función `ptx_link_stats_is_fresh` recibe como parámetro la variable `stats`, que apunta a las estadísticas para la tupla correspondiente.

4.3.4 Modificaciones en el módulo rpl-lite

Los cambios en rpl-lite están orientados a controlar la potencia de transmisión utilizada para enviar mensajes DIO y adaptar la elección del nodo a sondear teniendo en cuenta la potencia de transmisión.

- Control del nivel de potencia de los DIO. Los DIO son enviados por la función `rpl_icmp6_dio` desde el módulo icmpv6. Se ha modificado esta función para agregar la potencia como un nuevo parámetro de entrada. Antes de enviar el DIO, icmpv6 lee las variables relacionadas con la potencia del paquete para establecer el nivel de potencia deseado. Del mismo modo, se ha modificado la función que envía DIS para usar siempre el nivel de potencia predeterminado (máximo nivel de potencia).
- Envío de DIO multicast de forma periódica. Los paquetes DIO multicast se generan periódicamente según el algoritmo de goteo (Tricke), cambiando el nivel de potencia cada vez. Para esto, se ha agregado una nueva variable a la estructura de datos de información del DAG, `rpl_dag_t`, llamada `dio_next_ptx`, que indica la potencia de transmisión deseada y se cambia después de cada transmisión. El cambio de la variable es cíclico: el primer DIO multicast se envía a máxima potencia, el siguiente

a una potencia un nivel inferior, y así sucesivamente hasta la potencia mínima, después de la cual se vuelve a empezar por la potencia máxima.

- Modificación de la tabla de vecinos RPL. A la estructura **struct** rpl_nbr, que contiene la información para cada vecino, se ha añadido un campo de nombre better_ptx que contiene la potencia de transmisión óptima para ese vecino.
- Envío de DIO unicast (sondeo). Las modificaciones en el sondeo están orientadas a calcular la tupla [dirección mac, nivel de potencia] como objetivo potencial en lugar de sólo la dirección mac. Estas modificaciones también afectan a cómo se calcula el indicador de frescura, además de la función encargada de seleccionar el siguiente objetivo a sondear (get_probing_target). También se ha acortado el número de tuplas como objetivo potencial de sondeo, de acuerdo con el algoritmo 2, descartando como candidatos aquellos niveles de potencia mayores del necesario para mejorar la función objetivo de un nodo. En el código se utiliza la variable starting_index dentro de la función, para almacenar el valor equivalente a p_{max} del Algoritmo 2, y se puede ver en Código 4.4 un fragmento de la función get_probing_target . En este código se supone que nbr apunta al vecino para el que se está calculando la potencia a la que hay que sondear, y stats apunta a la tabla de NUM_PTX_LVLS elementos con sus estadísticas de enlace. El cálculo starting_index de la línea 5 es equivalente al cálculo de p_{max} en las líneas 14 a 18 del Algoritmo 2. Hay que tener en cuenta que en el código en C, los niveles de potencia están en el intervalo [0, NUM_PTX_LVLS] y en cambio en el Algoritmo 2 están en el intervalo [1,P]. Por eso, el bucle for de la línea 9 es equivalente al bucle for de la línea 19 del Algoritmo 2.

```
1
2   struct link_stats *stats = rpl_neighbor_get_link_stats (nbr);
3   int opt_ptx_index = get_ptx_lvls_index (nbr->better_ptx);
4
5   int starting_index = (opt_ptx_index > 0 &&
6                         stats->ptx_stats[opt_ptx_index].etx > 1) ?
7                        opt_ptx_index - 1 : opt_ptx_index;
8
9   for (int i = starting_index ; i < NUM_PTX_LVLS; i++){
```

Código 4.4 Fragmento de rpl-timers.c: cálculo de starting_index.

4.3.5 Función objetivo

La implementación de la función objetivo se basa en MRHOF y se puede resumir en los siguientes puntos:

- Implementación de métricas de enlace. Contiki-NG usa la función nbr_link_metric para calcular la métrica de enlace para un vecino. Se ha editado esta función para calcular la mejor métrica local de enlace para un vecino específico n como se indica en la ecuación 4.3, que refleja lo que se vio en el algoritmo 1.

$$nbr\_link\_metric = \min\left(ETX_n(p) \cdot P_{TX}(p)\right), \forall p \in [1:P] \tag{4.3}$$

También mediante esta función se almacena la mejor configuración de energía, es decir, el mejor nivel de potencia (campo better_ptx de la tabla de vecinos RPL) para ese vecino.

- Actualización del contenedor de métricas MC (Metric container). Se ha modificado la estructura de datos rpl_metric_container_t para incluir la mejor métrica de enlace. Después de recibir un DIO de un vecino, el contenedor de métricas se actualiza teniendo en cuenta el padre preferido.
- Ajustes de potencia de transmisión. Cada vez que se actualiza el estado del DAG con un padre preferido, se actualiza la variable global que controla el nivel de potencia de transmisión.

4.4 Conclusiones del capítulo

En este capítulo se ha descrito la implementación real de los dos algoritmos propuestos, modificando el sistema operativo Contiki-NG. Para poder trabajar con múltiples niveles de potencia se ha definido la constante NUM_PTX_LVLS y se ha modificado la tabla de estadísticas de enlace para que tenga una tabla con hasta NUM_PTX_LVLS elementos para cada vecino. También se ha modificado la tabla de vecinos RPL para almacenar la potencia óptima de transmisión para cada vecino. Se han modificado todas las funciones relacionadas con estas tablas y se han añadido nuevas funciones para manejar las múltiples potencias. Las modificaciones se han realizado intentando minimizar el número de líneas de código, para no afectar al consumo de energía y a la capacidad de cómputo, y también se ha intentado minimizar la memoria ocupada por las estructuras de datos, para ser compatibles con el mayor número de motas. La implementación realizada se encuentra disponible para la comunidad científica en [85]. El rendimiento de esta implementación será evaluado en los capítulos posteriores.

5 Pruebas de rendimiento y validación de la propuesta

En este capítulo se van a detallar las pruebas realizadas, explicando los motivos, los datos recogidos, las condiciones y la automatización.

5.1 Motivación

En apartados anteriores se han propuesto algoritmos que pretenden disminuir el consumo de energía, pero manteniendo un equilibrio respecto a sus posibles efectos sobre la calidad de servicio y el rendimiento del protocolo RPL. Se prevé que el resultado dependerá de la densidad de motas (inversamente proporcional a la distancia entre las motas) y por lo tanto en este apartado se pretende responder a cómo afecta esa densidad a las siguientes cuestiones:

1. Consumo de energía
2. Calidad de servicio
3. Rendimiento de RPL
4. Influencia del algoritmo de Probing alternativo

Para mostrar la influencia de la densidad de las motas, se realiza una batería de simulaciones con un escenario donde se varía esa densidad, y se recolectan los estadísticos necesarios para responder a las anteriores cuestiones. Estos estadísticos que se recolectan se pueden agrupar en:

- Energía: consumo total y consumo desglosado.
- Tráfico y calidad de servicio: retardo extremo a extremo, contadores (mensajes de datos, tramas enviadas, tramas no asentidas, retransmisiones, tramas recibidas).
- RPL: Mensajes de control, cambio de padre y retardo en unión al árbol (JoinedDelay).

y en el siguiente apartado se especifican con más detalle.

Tabla 5.1 Principales métricas recogidas en las simulaciones.

	Indicador	Descripción
Energía	Energía *Total* (mJ)	consumo de energía conjunto (CPU + radio)
	Energía CPU_{act} (mJ)	consumo de energía por la CPU en modo activo
	Energía CPU_{LPM} (mJ)	consumo de energía por la CPU en modo LPM
	Energía $RADIO_{IDLE}$ (mJ)	consumo de energía por la radio en modo IDLE
	Energía $RADIO_{RX}$ (mJ)	consumo de energía por la radio en recepción
	Energía $RADIO_{TX}$ (mJ)	consumo de energía por la radio en transmisión
Tráfico y QoS	APP-*i*	# paquetes de datos enviados y recibidos con éxito
	Tramas asentidas	# tramas unicast enviadas con éxito
	Tramas no asentidas	# tramas unicast enviadas sin éxito
	Retransmisiones	# retransmisiones
	Retardo (ms)	retardo medio extremo a extremo para los paquetes de datos
RPL	U-DIO-*i*	# unicast DIO enviados
	M-DIO-*i*	# multicast DIO enviados
	Cambios de padre	# veces que un nodo cambia su padre preferido
	Retardo de unión (s)	retardo medio en unirse al DODAG

5.2 Colección de estadísticos

Los estadísticos recolectados se recopilan en distintas métricas, que reflejan el funcionamiento de los esquemas propuestos. Estas métricas recopiladas se enumeran en la Tabla 5.1, donde el índice *i* indica el nivel de potencia de transmisión (por ejemplo, APP-*i* indica el número de paquetes de datos transmitidos utilizando el nivel de potencia *i*).

La energía consumida en cada nodo se calcula como se indica en la ecuación 5.1, donde la tensión es $V = 3.2V$, las intensidades se obtienen de las hojas de especificaciones y los tiempos se deben recopilar mediante pruebas.

$$\begin{aligned} E_{node} =& E_{CPU\_TOTAL} + E_{RADIO} = \\ =& E_{CPU} + E_{LPM} + E_{IDLE} + E_{TX} + E_{RX} = \\ =& V \cdot (T_{CPU} \cdot I_{CPU} + T_{LPM} \cdot I_{LPM} + T_{IDLE} \cdot I_{IDLE} + \\ &+ \sum_{i=1}^{P} T_{TX}(i) \cdot I_{TX}(i) + T_{RX} \cdot I_{RX}) \end{aligned} \tag{5.1}$$

De la hoja de especificaciones de la radio CC2420[84] se obtienen los valores de las intensidades de la radio en sus distintos estados *IDLE*, *TX* y *RX*:

- $I_{IDLE} = 0.426mA$
- $I_{RX} = 18.8mA$
- $I_{TX}(i)$ es la intensidad para el nivel de potencia de transmisión *i* (ver Tabla 5.5) y varía entre $9.9mA$ y $17.4mA$

De la hoja de especificaciones de la CPU MSP430f2617 [86] se obtienen los valores de las intensidades de la CPU en sus distintos estados *CPU* (modo activo) y *LPM* (modo bajo consumo):

- $I_{CPU} = 4.5mA$ se obtiene de la gráfica de "Figure 8-2. Active Mode Current vs Supply Voltage (TA = 25°C)" de esa misma hoja de especificaciones de la CPU, considerando que la mayor parte del tiempo trabaja a 8 MHz y la tensión de alimentación es $3.2V$.
- $I_{LPM} = 0.0005mA$

Los tiempos que se capturan para el cómputo de la energía son los siguientes:

- T_{CPU} es el tiempo en el que la CPU está en modo activo
- T_{LPM} es el tiempo en el que la CPU está en modo LPM
- T_{IDLE} es el tiempo en el que la radio está en modo IDLE
- $T_{TX}(i)$ es el tiempo de transmisión para el nivel de potencia i
- T_{RX} es el tiempo de recepción

Para realizar pruebas y mediciones, se necesita una plataforma, y se va a utilizar el simulador Cooja [48], del que se hará una descripción en el apartado 5.3.

Para medir los tiempos y poder calcular las métricas de consumo de energía se usan dos complementos, Energest (de Contiki-ng), con ligeras modificaciones para distinguir entre niveles de potencia y PowerTracker (de Cooja).

Para recopilar las métricas relacionadas con el tráfico y calidad de servicio, y también con RPL, se utiliza el complemento Event-count, desarrollado en [80] para usarlo en Contiki-ng.

Estas herramientas forman parte del código compilado en cada mota, de manera que irán recopilando la información almacenándola en variables, y en el momento que se les indique, justo antes de terminar la simulación, volcarán el contenido de todas las variables por la salida estándar de la mota (utilizando *printf*).

El complemento 'Simulation Script Editor' se usa para realizar un programa en JavaScript que lee los datos generados por las motas, y los escribe en un fichero de resultados.

5.3 Plataforma de simulación y complementos

Cooja es un simulador basado en Java, diseñado para la simulación de redes de sensores con el sistema operativo Contiki o Contiki-NG. El simulador está implementado en Java, pero permite la modificación y compilación de los programas que se ejecutan en las motas, que están escritos en lenguaje C. Este simulador de red, de código abierto, se ha convertido en una herramienta ampliamente utilizada en el dominio de redes de sensores inalámbricos y aplicaciones del IoT [87, 88]. Cooja también se usa ampliamente para estudiar el rendimiento de RPL [89, 90] (incluido xRPL [80]), ya que incorpora complementos (también llamados módulos o plugins) especializados, que interactúan con el código de los nodos, y le permiten la recopilación de métricas relacionadas con el consumo de energía, la calidad de servicio y el rendimiento de RPL.

Cooja permite compilar y simular las aplicaciones en Contiki-NG con sus respectivos ficheros ubicados en un directorio independiente. En ese directorio, el fichero project−conf.h, permite la configuración de distintos parámetros de simulación antes de la compilación.

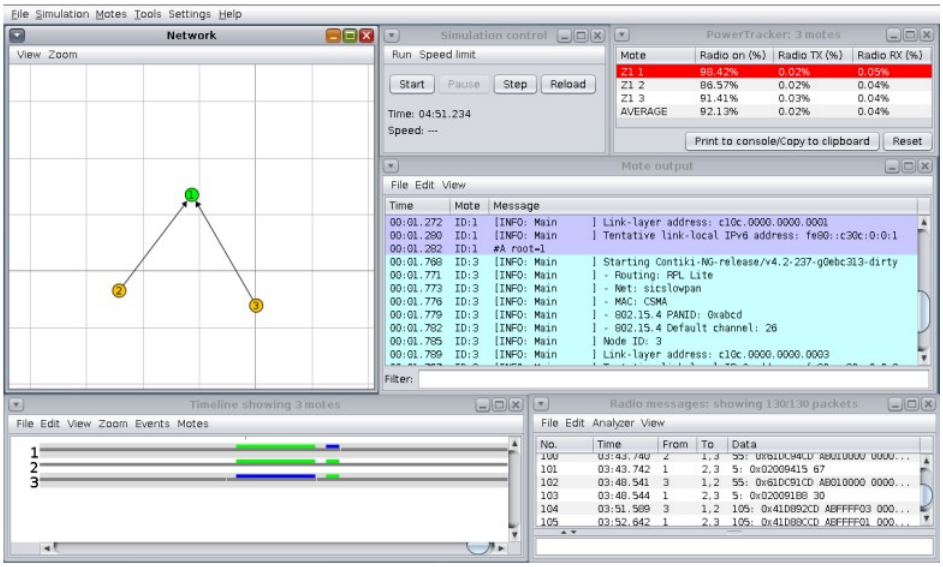


Figura 5.1 Interfaz de usuario de Cooja.

Este fichero permite, por ejemplo, configurar RPL, la capa MAC, o los distintos complementos usados en la simulación.

Este simulador puede funcionar utilizando una interfaz gráfica de usuario (GUI), como se puede ver en la Figura5.1, en la que aparecen diferentes ventanas con las que se pueden interactuar y ver la información acerca de la simulación. La ejecución del simulador se inicia mediante la orden ant run. También puede funcionar a través de la línea de comando, sin interfaz gráfica de usuario, usando la orden ant run_nogui −Dargs=<fichero.csc>. Para ambos métodos, la configuración de una simulación se almacena en un fichero XML con extensión ".csc" (Cooja Simulation Configuration), que contiene información acerca del entorno de simulación, complementos, las motas y su posición, la semilla para la aleatoriedad, las propiedades del medio radio, etc. Como ejemplo, se muestra un fragmento de un fichero de este tipo en Código 5.1.

```
<?xml version="1.0" encoding="UTF−8"?>
<simconf>
  <project EXPORT="discard">[APPS_DIR]/mrm</project>
  <project EXPORT="discard">[APPS_DIR]/mspsim</project>
  <project EXPORT="discard">[APPS_DIR]/avrora</project>
  <project EXPORT="discard">[APPS_DIR]/serial_socket</project>
  <project EXPORT="discard">[APPS_DIR]/powertracker</project>
  <simulation>
    < title >Escenario aleatorio </ title >
    <randomseed>4045679908202524122</randomseed>
    <motedelay_us>1000000</motedelay_us>
    <radiomedium>
      org. contikios .cooja .radiomediums.UDGM
      < transmitting_range >50.0</ transmitting_range >
```

```
    < interference_range >100.0</ interference_range >
    < success_ratio_tx >1.0</ success_ratio_tx >
    < success_ratio_rx >1.0</ success_ratio_rx >
  </radiomedium>
  <events>
    <logoutput>40000</logoutput>
  </events>
  <motetype>
    org. contikios .cooja.mspmote.Z1MoteType
    < identifier >z11</ identifier >
    < description >Z1 Mote Server</ description >
```

Código 5.1 Fragmento de fichero de tipo ".csc".

En Cooja, la interacción con las motas simuladas se realiza mediante complementos con diferentes funcionalidades. Por ejemplo, el complemento PowerTracker permite conocer cuánto tiempo, y porcentaje de tiempo, han permanecido las radios de las motas en sus diferentes estados (transmitiendo, recibiendo y encendida). También existe otro complemento, llamado 'Simulation Script Editor', que permite usar un código JavaScript que será ejecutado durante la simulación, interactuando directamente con la salida estándar de las motas (printf). Además, el código JavaScript de este complemento puede acceder a la información del resto de complementos de la simulación y, al final de la simulación, crear un fichero con toda la información generada (COOJA.testlog). Pero la característica más importante de este complemento es que con él se pueden temporizar las simulaciones, siendo ésta la única forma que ofrece Cooja. Otros complementos que se va a usar son Energest y Event-count.

5.3.1 Complemento Energest

El complemento Energest de Contiki-ng es el que se va a usar (junto con PowerTracker de Cooja) para coleccionar los estadísticos relacionados con la energía. Energest [91] es un complemento que se usa en Contiki-NG para contabilizar el tiempo que permanecen encendidos, apagados, o en otros estados los diferentes componentes, como la CPU y la radio. De esta manera, una vez que se conoce el tiempo que un componente ha estado encendido, se puede estimar la energía consumida en un dispositivo o mota con el cálculo que se vio en la ecuación 5.1. Este complemento ofrece una serie de funciones y macros a través de una API, mostrada en la Tabla 5.2. De todas las funciones y macros, las que

Tabla 5.2 API del complemento Energest.

Funciones y macros	Propósito
ENERGEST_TIME_T ENERGEST_CURRENT_TIME()	Get the system time from the Energest time source.
ENERGEST_SECOND	The number of ticks per second from the Energest time source.
ENERGEST_ON(type)	Set the specified type to on and start tracking the time.
ENERGEST_OFF(type)	Set the specified type to off and update the total time.
ENERGEST_SWITCH(type_off, type_on)	Switch from tracking time for the first type to the second type.
ENERGEST_GET_TOTAL_TIME()	Get the total time Energest has been tracking.
void energest_flush(void)	Flush the Energest times for all components that are currently turned on.
uint64_t energest_type_time(energest_type_t type)	Read the total time the specified type has been turned on.
void energest_init(void)	Initialize the Energest module.

están relacionadas con esta tesis son:

- ENERGEST_SECOND: Como Energest cuenta en ticks (reloj de la CPU), el número de ticks por segundo se usa para pasar el resultado de los contadores a segundos.
- ENERGEST_ON y ENERGEST_OFF: Ya que, aunque se utilicen las variables que vienen por defecto en el complemento, hay que añadir algunas nuevas para contabilizar los tiempos de transmisión a diferentes potencias.
- energest_flush : Antes de obtener los tiempos y presentarlos por la salida estándar, deben ser actualizados.
- energest_type_time : Para obtener los tiempos de cada variable y poder presentarlos por la salida estándar.
- energest_init : Para poder iniciar el complemento en las motas.

Por otro lado, los tipos de variables que contabilizan el tiempo de la CPU y la radio, y que vienen por defecto en el complemento son los siguientes:

- ENERGEST_TYPE_CPU: Almacena el tiempo que la CPU ha estado en modo activo.
- ENERGEST_TYPE_LPM: Almacena el tiempo que la CPU ha estado en modo de baja potencia.
- ENERGEST_TYPE_TRANSMIT: Almacena el tiempo que la radio ha estado transmitiendo.
- ENERGEST_TYPE_LISTEN: Almacena el tiempo que la radio ha estado escuchando y recibiendo, es decir, el tiempo que no ha estado transmitiendo. Energest no diferencia entre escuchando y recibiendo, ya que la radio sólo indica cuándo se ha recibido una trama y no el tiempo que ha tardado en recibirla.

Debido a que además se quiere diferenciar entre el tiempo de transmisión a distintas potencias, para realizar los cálculos de energía consumida, esta herramienta requiere modificaciones para poder ofrecer esta información. Los cambios realizados en el complemento Energest son los siguientes:

- Introducción de nuevos valores:
  - ENERGEST_TYPE_TRANSMIT_PTX1
  - ENERGEST_TYPE_TRANSMIT_PTX2
  - etc.

  para contabilizar los tiempos de transmisión a distintas potencias. Esto se hace configurando el parámetro ENERGEST_CONF_PLATFORM_ADDITIONS en el fichero project−conf.h, tal como se muestra en el Código 5.2.

```
#define ENERGEST_CONF_PLATFORM_ADDITIONS
        ENERGEST_TYPE_TRANSMIT_PTX1, ENERGEST_TYPE_TRANSMIT_PTX2,
        ENERGEST_TYPE_TRANSMIT_PTX3, ENERGEST_TYPE_TRANSMIT_PTX4,
        ENERGEST_TYPE_TRANSMIT_PTX5, ENERGEST_TYPE_TRANSMIT_PTX6,
        ENERGEST_TYPE_TRANSMIT_PTX7, ENERGEST_TYPE_TRANSMIT_PTX8
```

Código 5.2 Fragmento de project-conf.h.

Para poder utilizar esos valores como valores numéricos, se ha modificado el fichero energest.h, como se puede observar en el Código 5.3.

```
typedef enum energest_type {
  ENERGEST_TYPE_CPU,
  ENERGEST_TYPE_LPM,
  ENERGEST_TYPE_DEEP_LPM,
  ENERGEST_TYPE_TRANSMIT,
  ENERGEST_TYPE_LISTEN,
#ifdef ENERGEST_CONF_PLATFORM_ADDITIONS
  ENERGEST_CONF_PLATFORM_ADDITIONS,
#endif /* ENERGEST_CONF_PLATFORM_ADDITIONS */
  ENERGEST_TYPE_MAX
} energest_type_t;
```

Código 5.3 Fragmento de energest.h.

Estos valores tendrán entonces asociados un valor numérico, donde se cumplirá que ENERGEST_TYPE_TRANSMIT_PTX2 = ENERGEST_TYPE_TRANSMIT_PTX1 + 1 y ENERGEST_TYPE_TRANSMIT_PTX8 = ENERGEST_TYPE_TRANSMIT_PTX1 + 7, por ser un tipo enumerado.

- Sustitución, dentro del código que gestiona la radio, el código que permite contar el tiempo de transmisión haciendo uso de ENERGEST_ON y ENERGEST_OFF aplicado a ENERGEST_TYPE_TRANSMIT, por algo un poco más complejo que permite obtener, antes del ENERGEST_ON, la potencia de transmisión y así poder contabilizar las distintas potencias.
- Uso de los nuevos valores en la función que imprime los contadores de Energest dentro del código que se ejecuta en las motas, tanto en el servidor (nodo raíz), como en los clientes (resto de las notas). El código de los clientes se puede ver en Código 5.4.

```
static void
print_energest (void *ptr)
{
  energest_flush ();
  printf ("Energest: ENERGEST_SECOND:%u-CPU:%lu-LPM:%lu-LISTEN:%lu",
          ENERGEST_SECOND,
          (unsigned long) energest_type_time (ENERGEST_TYPE_CPU),
          (unsigned long) energest_type_time (ENERGEST_TYPE_LPM),
          (unsigned long) energest_type_time (ENERGEST_TYPE_LISTEN));
  char c='0';
  int i=0;
  for (;i<NUM_PTX_LVLS;i++){
    c++;
    printf ("-PTX%c:%lu", c,
```

```
            (unsigned long) energest_type_time (ENERGEST_TYPE_TRANSMIT_PTX1+i));
    }
    printf ("-TOTAL_TIME:%lu\n",
            (unsigned long)ENERGEST_GET_TOTAL_TIME());
}
```

Código 5.4 Fragmento de udp-client.c con llamada a print_energest ().

Esta función print_energest () se ejecuta al final de la simulación, y consigque que la mota imprima de una vez todos los valores calculados. El uso del bucle **for** con la variable *i* y la constante NUM_PTX_LVLS hace que el código no dependa del número de niveles de potencia que se quieran simular (por ejemplo, NUM_PTX_LVLS 5).

Para finalizar, hay que indicar que la inicialización y obtención de los tiempos contabilizados por el complemento Energest se realiza a nivel de aplicación, por este motivo en la Figura 5.2 aparece la inicialización del complemento. También hay que decir que a la hora de exportar los tiempos por la salida estándar se ha decidido que estén medidos en ticks y no en segundos, ofreciendo también el valor de ENERGEST_SECOND para que se puedan calcular los segundos.

5.3.2 Complemento PowerTracker

PowerTracker es un plugin o complemento de Cooja que permite conocer el tiempo que permanece la radio encendida, transmitiendo y recibiendo. A diferencia de Energest, PowerTracker ofrece el tiempo real recibiendo T_{RX}, por lo que capturando este valor (que se obtiene a través del complemento 'Simulation Script Editor') y con el valor ENERGEST_TYPE_LISTEN (que se obtiene de Energest), como se indica en la ecuación 5.2, se puede calcular el tiempo de IDLE como la resta de ambos y tener así el tiempo de IDLE y RX por separado.

$$\begin{aligned} T_{LISTEN} &= (\text{se obtiene de Energest}) \\ T_{RX} &= (\text{se obtiene de PowerTracker}) \\ T_{IDLE} &= T_{LISTEN} - T_{RX} \end{aligned} \tag{5.2}$$

5.3.3 Complemento Event-count

El complemento Event-count, desarrollado en [80] para usarlo en Contiki-ng, se usa para recopilar las métricas relacionadas con el tráfico y calidad de servicio, y también con RPL. Su función es contabilizar diferentes eventos que ocurren durante la ejecución del código, como el envío de los diferentes tipos de paquetes, las veces que una mota cambia de padre o el número de paquetes recibidos. Este complemento es relativamente sencillo y tiene una estructura inspirada en la del complemento Energest: va recopilando información durante la simulación para que al final de la simulación se envíe toda esa información en un mensaje por la salida estándar (printf). Antes de usar este complemento se intentó enviar un mensaje a la salida estándar cada vez que ocurría un evento, pero eso hacía un uso excesivo de printf , consumiendo muchos recursos y afectando negativamente a las

simulaciones. No obstante, printf se usa durante la simulación cada vez que una mota envía un paquete de datos, o cada vez que la mota raíz recibe un paquete de datos (para calcular los retardos), pero eso ocurre menos veces. Al igual que Energest, ofrece una serie de funciones y macros a través de una API, mostrada en la Tabla 5.3. Con las funciones

Tabla 5.3 API del complemento Event-count.

Funciones y macros	Propósito
EVENT_COUNT(type)	Avanza la cuenta en uno de la variable type.
EVENT_COUNT_NUM(type,num)	Avanza la cuenta según indique num de la variable type.
EVENT_COUNT_GET(type)	Obtiene la cuenta de la variable type.
event_count_print_all_events(void)	Imprime por la salida estándar el listado de variables con sus valores actuales.
void event_count_reset_all(void)	Reinicia todas las variables
event_count_init(void)	Inicia el complemento Event-count.

EVENT_COUNT y EVENT_COUNT_NUM se contabiliza el evento que se desee, y se utiliza event_count_print_all_events para que la mota imprima los resultados. El listado de variables que tiene el complemento es el siguiente:

- EVENT_TYPE_SEND_MSG: Evento de envío de un paquete a nivel APP.
- EVENT_TYPE_SEND_MSG_*i*: Evento de envío de un paquete a nivel APP a potencia *i*.
- EVENT_TYPE_SEND_M_DIO: Evento del envío de mensaje DIO multicast.
- EVENT_TYPE_SEND_M_DIO_*i*: Evento del envío de mensaje DIO multicast a potencia *i*.
- EVENT_TYPE_SEND_U_DIO: Evento del envío de mensaje DIO unicast.
- EVENT_TYPE_SEND_U_DIO_*i*: Evento del envío de mensaje DIO unicast a potencia *i*.
- EVENT_TYPE_SEND_PACKET: Evento de envío de cualquier paquete a nivel MAC.
- EVENT_TYPE_PARENT_SWITCH: Evento de cambio de padre preferido.
- EVENT_TYPE_TX_OK: Evento de paquete transmitido correctamente a nivel MAC.
- EVENT_TYPE_TX_NO_ACK: Evento de paquete no enviado debido a que no llegó el ACK.
- EVENT_TYPE_NUM_RTX: Evento que cuenta cada vez que se retransmite un paquete.

Para finalizar, decir que, al igual que Energest, la inicialización y obtención de los eventos contabilizados se realiza a nivel de aplicación, por este motivo en la Figura 5.2 aparece la inicialización del complemento. También decir que el uso de la función event_count_print_all_events () para exportar los valores contabilizados a través de la salida estándar se realizará al final de la simulación, y estos valores serán recogidos por el

complemento 'Simulation Script Editor'. El código de los clientes donde se ejecuta la llamada a esa función (igualmente en el servidor) se muestra en Código 5.5. En este código se observa que cuando una mota recibe el carácter 't', entonces llama a la función event_count_print_all_events (). También se observa la llamada a la función print_energest () mostrada en Código 5.4.

```
static int uart_rx_callback (unsigned char c) {
  if(c == 't'){
    event_count_print_all_events ();
    print_energest (NULL);
  }
  return 0;
}
```

Código 5.5 Fragmento de udp-client.c con llamada a event_count_print_all_events ().

5.3.4 Complemento 'Simulation Script Editor'

Este complemento de Cooja (escrito en JavaScript) permite interactuar con la simulación. Este código JavaScript puede cargar el resto de plugins que se utilizan durante la simulación y obtener la información que generan. También puede acceder directamente a la entrada/salida estándar de las motas en tiempo de simulación, es decir, en cuanto se genera el mensaje en la salida, sin necesidad de cargar el complemento 'LogListener'.

Aunque esta herramienta está pensada para hacer pruebas de las simulaciones, en esta tesis se utilizará únicamente para comunicarse con las motas, recabar información de la simulación, detenerla, y escribir en un fichero los resultados de una simulación. El fichero generado por el código JavaScript tiene el nombre COOJA.testlog y es un fichero de texto que contiene lo generado por la función log.log(). Este fichero contiene los resultados básicos de una simulación en formato de texto, y se procesa posteriormente mediante un script en Python para generar una tabla en formato csv.

El código JavaScript usado en esta tesis se ha almacenado en el fichero simul_sc.js, y su funcionalidad se puede dividir en las siguientes partes:

1. Programa la finalización de la simulación (por ejemplo 10 horas ó 36000000 ms).
2. Lee los mensajes enviados por las motas a la salida estándar durante toda la simulación, para calcular el retardo extremo a extremo de los paquetes de datos y también el retardo medio en unirse al DODAG. Estos mensajes pueden ser:
   - DATA sent:. Estos mensajes los envían las motas cada vez que envían un paquete de datos al nodo raíz. Se almacena el instante de envío.
   - DATA received:. Estos mensajes los envía la mota raíz cada vez que recibe un paquete de datos. Con el instante de recepción y el instante de envío previamente almacenado, se calcula el retardo de un paquete de datos y se van acumulando los retardos y contando los paquetes. Al final de la simulación se hará la media de retardos por cada mota, y posteriormente la media de todas las motas: $Retardo(ms)$.

- Node joined. Estos mensajes los envían las motas al unirse al DODAG. Se almacena el instante de envío del mensaje, un mensaje para cada mota, e indica el tiempo que ha tardado la mota en unirse. Al final de la simulación se hará la media de todas las motas: *R*etardo de unión (*s*).

3. Al finalizar la simulación escribe en el fichero COOJA.testlog la información correspondiente a cada mota que se ha ido recogiendo de la salida estándar durante toda la simulación:
   - Retardo acumulado de los paquetes de datos
   - Contador de paquetes de datos
   - Tiempo que ha tardado la mota en unirse al DODAG
   - Información recopilada por el complemento PowerTracker, dentro de la cual está el tiempo que ha estado la mota recibiendo datos T_{RX}
4. Envía un mensaje a cada mota cuyo contenido es el carácter 't', para que éstas envíen a la salida estándar toda la información recolectada durante la simulación con los complementos Event-count y Energest. Este mensaje es el que llega a la función que se muestra en Código 5.5.
5. Espera a que cada mota envíe a la salida estándar 2 mensajes, uno de Event-count y otro de Energest, los escribe en el fichero COOJA.testlog, y termina.

5.3.5 Resumen de complementos

A modo de resumen, la información recogida en el fichero COOJA.testlog para cada mota es:

- Retardo acumulado de los paquetes de datos
- Contador de paquetes de datos
- Tiempo que ha tardado la mota en unirse al DODAG
- T_{RX}
- Mensaje de Event-count
- Mensaje de Energest

Se recuerda que la razón de que Event-count (y también Energest) envíen a la salida estándar toda la información en un mensaje al final de la simulación, es para minimizar el uso de printf por parte de la mota. También se recuerda que se ha usado el complemento PowerTracker de Cooja para obtener T_{RX} y poder calcular $T_{IDLE} = T_{LISTEN} - T_{RX}$.

La información recogida se va a procesar posteriormente para generar las métricas especificadas en la Tabla 5.1, y se puede ver cómo se usa cada complemento para obtener cada métrica en la Tabla 5.4.

Una vez que se ha especificado cómo usar el simulador Cooja (con sus complementos PowerTRacker y 'Simulation Script Editor') y los complementos de Contiki-ng (Energest y Event-count), y sabiendo que el resultado de una simulación se guarda en el fichero COOJA.testlog, se procede a explicar cómo es el escenario base de una simulación y la automatización de las simulaciones.

Tabla 5.4 Complementos usados para las métricas.

	Métrica	Energest	PowerTracker	Event-count	Simulation Script Editor
Energía	Energía $TOTAL$ (mJ)	X			
	Energía CPU_{act} (mJ)	X			
	Energía $RADIO_{IDLE}$ (mJ)	X			
	Energía CPU_{LPM} (mJ)	X			
	Energía $RADIO_{RX}$ (mJ)	X	X		
	Energía $RADIO_{TX}$ (mJ)	X			
Tráfico y QoS	APP-i			X	
	Tramas asentidas			X	
	Tramas no asentidas			X	
	Retransmisiones			X	
	Retardo (ms)				X
RPL	U-DIO-i			X	
	M-DIO-i			X	
	Cambios de padre			X	
	Retardo de unión (s)				X

5.4 Escenario base de simulación

En los siguientes apartados se mostrarán los aspectos comunes a todas las simulaciones.

5.4.1 Hardware del nodo: mota Z1

Una característica necesaria a la hora de elegir una mota con la que hacer las simulaciones es que la radio de la mota pueda transmitir a diferentes potencias. Con esa característica existen distintas motas en Contiki-NG (es decir, que implementan el sistema operativo), como, por ejemplo, CC2650 TI, RE Mote (Zolertia), Sky o Z1 (Zolertia). Pero, aunque en Contiki-NG existan múltiples motas, no todas ellas están implementadas en el simulador Cooja, siendo las motas Sky y Z1 las únicas disponibles en Cooja que permiten trabajar con diferentes potencias. Además, estas motas, junto con la mota genérica de Cooja son las más utilizadas por la comunidad para hacer simulaciones con Cooja.

Las motas Sky y Z1 son muy parecidas, ya que poseen la misma radio (CC2420) y un microcontrolador de la misma familia (MSP430). La mayor diferencia entre estas motas es la memoria Flash (ROM), teniendo la de la mota Z1 aproximadamente el doble de capacidad que la mota Sky. Este último factor hace preferible la mota Z1, ya que las modificaciones para implementar el algoritmo de capa cruzada requieren de más memoria ROM, siendo la memoria Flash de la mota Sky insuficiente. Por otro lado, la modificación del algoritmo también requiere de más memoria RAM, y aunque la mota Sky tenga más, en las simulaciones con la mota Z1 no se ha tenido ningún problema en este aspecto. Como conclusión, se tiene que la única mota disponible para poder hacer simulaciones en Cooja transmitiendo a diferentes potencias y que tenga la suficiente capacidad para almacenar las modificaciones es la mota Z1 de Zolertia, y aunque sea una mota de una cierta antigüedad, es ampliamente citada en la literatura [92, 88].

Las características generales más importantes de la mota Z1 son las siguientes:

- Microcontrolador MSP430F217 de bajo consumo.

- CPU con una velocidad de hasta 16MHz.
- Memoria RAM de 8KB y memoria Flash de 92 KB.
- Transceptor radio CC2420.
- Soporte con el estándar IEEE 802.15.4
- Operabilidad a 2.4Ghz con una tasa de 250Kbps.

Aunque el transceptor radio de la mota tiene 8 niveles de potencia, tal como se mostró en la Figura 4.3, en esta tesis se han usado 5 niveles en las simulaciones, suficientes para mostrar los beneficios del uso de múltiples niveles de potencia, pero sin aumentar en exceso el uso de memoria y la complejidad de cómputo. En la Tabla 5.5, se muestran los niveles de potencia seleccionados para MxRPL. Para xRPL, con sólo dos niveles de potencia, se usa $H = 1$ como nivel de potencia alto y $L = 5$ como nivel bajo (es decir, $P_{TX}(H) = 55$ mW y $P_{TX}(L) = 31$ mW).

Tabla 5.5 Niveles de potencia de transmisión en mota Z1.

Nivel	Potencia (dBm)	Potencia (mW)	Intensidad (mA)
$P_{TX}(1)$	0	55	17.4
$P_{TX}(2)$	-3	48	15.2
$P_{TX}(3)$	-5	44	13.9
$P_{TX}(4)$	-10	35	11.2
$P_{TX}(5)$	-15	31	9.9

5.4.2 Disposición de motas y tamaño del terreno

Como los resultados dependen previsiblemente de la densidad de motas, es decir, de la cercanía entre ellas, se van a hacer distintas simulaciones donde la densidad vaya variando. Para ello se va a utilizar un número fijo de motas (15 nodos además del raíz) diseminadas en una superficie cuadrada de diferentes tamaños (desde $10m \times 10m$ hasta $100m \times 100m$). Este rango de tamaño permite que las motas utilicen su gama completa de niveles de potencia de transmisión. Si bien los 15 nodos se distribuyen aleatoriamente en cada simulación, el nodo raíz siempre está en el centro del cuadrado. Este escenario permite una comparación directa con xRPL [80], donde se mostró que áreas mayores de $100m \times 100m$ podrían producir nodos desconectados.

5.4.3 Generación de tráfico

Los nodos ejecutan la aplicación de ejemplo de servidor udp y cliente udp predeterminada de Contiki-NG, donde los nodos envían un paquete de datos (mensaje "HELLO") al nodo raíz periódicamente. Se realizan simulaciones de 10 h de funcionamiento, con transmisiones de datos cada 10 s. Esto genera alrededor de 3600 paquetes de datos (cada nodo genera un total de 3597 paquetes de datos), suficiente para producir resultados confiables y reducir la influencia del tráfico de control en los resultados. A fin de evitar colisiones iniciales, los nodos envían su primer paquete aleatoriamente en los primeros 10 s del tiempo de simulación.

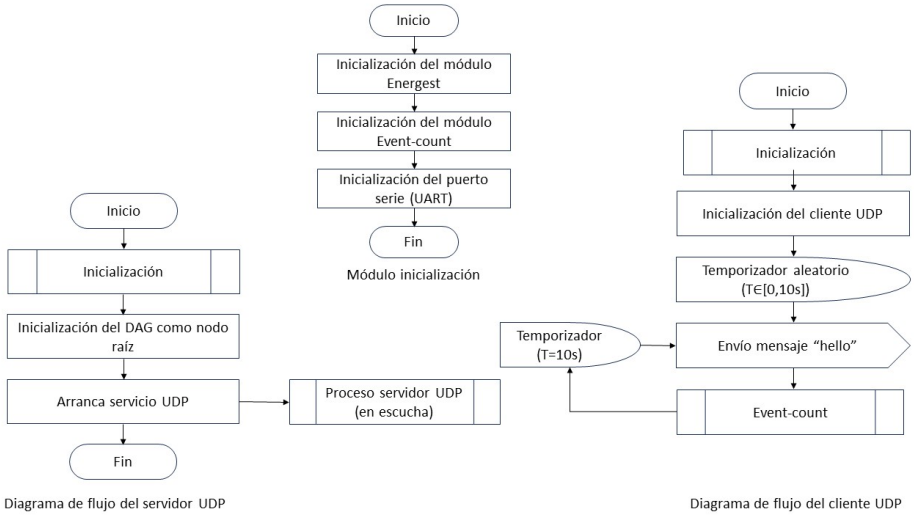


Figura 5.2 Diagramas de flujo de UDP.

Para realizar esta aplicación se ha programado de forma diferente el comportamiento de la mota raíz, que funcionará como un sumidero de tráfico, y del resto de motas, que serán las que generen los mensajes "HELLO" (paquetes de datos). Esta programación se ha llevado a cabo en los ficheros udp−server.c y udp−client.c que serán los que se compilen con el simulador Cooja, uno en la mota raíz y el otro en el resto motas respectivamente.

El funcionamiento de udp−server.c y udp−client.c se ve reflejado en los diagramas de flujo de la Figura 5.2. En ambos diagramas aparece la inicialización de Energest, Event-count y el puerto serie (UART), que se utilizan para la recogida de datos en la mota. La entrada/salida estándar está conectada con el puerto serie, y permite a las motas comunicarse con el simulador a través del complemento 'Simulation Script Editor'.

En el diagrama del servidor UDP (implementado en el nodo raíz) se inicializa el nodo raíz RPL y se habilita un proceso encargado de la recepción de los mensajes UDP. Este proceso sólo se encarga de la recepción de los paquetes sin procesar la información recibida, ya que el único objetivo de la aplicación es generar tráfico. Cada vez que recibe un paquete, envía un mensaje DATA received: a la salida estándar, indicando el emisor del paquete (el receptor será el raíz, que es el servidor).

En el diagrama del cliente UDP (implementado en todas las motas menos en el nodo raíz), además de realizar el envío de mensajes, también se están contabilizando el número de mensajes total enviados por cada nodo y el número de mensajes enviados en cada nivel de potencia. Para ello, se hace uso de una herramienta Event-count. Cada vez que envía un paquete, envía un mensaje DATA sent: a la salida estándar, indicando el emisor del paquete (el receptor será el raíz, que es el servidor).

Estos mensajes enviados a la salida estándar son los que permiten al simulador calcular el retardo de extremo a extremo, mediante el complemento 'Simulation Script Editor', que leerá esos mensajes y calculará los tiempos.

5.4.4 Modelo de propagación del canal radio

Se usa el modelo UDGM de pérdida con la distancia, en el que tanto RSSI como la probabilidad de error cambian linealmente con la distancia. Por lo tanto, cuanto mayor distancia, menor RSSI y mayor probabilidad de error en recepción. Se han deshabilitado lo errores en los eventos de transmisión y recepción para simplificar el análisis de resultados.

5.5 Condiciones de las pruebas

En este apartado se van a describir las condiciones de las pruebas realizadas, teniendo en cuenta que se quiere comparar el funcionamiento de los distintos esquemas, en escenarios donde se vaya variando las densidad de las motas.

5.5.1 Esquemas

A fin de comparar el rendimiento de la propuesta con otras existentes, se van a ejecutar las simulaciones con los distintos esquemas que se nombran a continuación:

- **RPL**: uso del protocolo estándar RPL.
- **xRPL**: adaptación de RPL para 2 niveles de potencia [80] (similar al Algoritmo 1 de capa cruzada pero sólo con 2 niveles).
- **MxRPL**: adaptación propuesta en esta tesis (especificada en el Algoritmo 1 de capa cruzada con 5 niveles de potencia). Respeta el algoritmo de Probing original.
- **MxRPL-AP**: propuesta completa descrita en esta tesis, incluyendo el Algoritmo 1 de capa cruzada con 5 niveles de potencia y el Algoritmo 2 de probing alternativo.

En las herramientas de automatización de las simulaciones se utilizará una variable de nombre *TIPO_RPL*, para indicar qué esquema es el que se está usando.

5.5.2 Tamaño del terreno (escenarios con distinta densidad de motas)

Para probar distintos escenarios donde la densidad de las motas sea distinta, se van a usar 8 escenarios con distinto tamaño del terreno, con 15 motas de tipo cliente distribuidas aleatoriamente (se cambian de posición en cada simulación) y además la mota raíz en el centro. El terreno va a ser una superficie cuadrada, donde los lados toman valores entre $10m$ y $100m$:

1. $10m \times 10m$
2. $20m \times 20m$
3. $30m \times 30m$
4. $40m \times 40m$
5. $50m \times 50m$
6. $60m \times 60m$

7. $80m \times 80m$
8. $100m \times 100m$

En las herramientas de automatización de las simulaciones se utilizará una variable de nombre *TAM_RED*, que contiene la longitud del lado del cuadrado (m), con valores desde $TAM\_RED = 010$ hasta $TAM\_RED = 100$, para indicar el tamaño del terreno que se está usando.

Cabe señalar que el tamaño máximo del terreno viene determinado por la necesidad de que ningún nodo quede aislado del árbol RPL. Para tamaños superiores a $100m \times 100m$ pudieron observarse nodos aislados en las simulaciones, lo que perturba la interpretación de los resultados obtenidos.

5.6 Automatización de las simulaciones

Una vez visto que el escenario de una simulación se especifica en un fichero XML con extensión .csc (que en esta tesis vamos a llamar escenario.csc), que su resultado se escribe en el fichero COOJA.testlog, y los aspectos comunes a todas las simulaciones, se describen a continuación las herramientas usadas para automatizar múltiples simulaciones:

- simulaciones.sh
- ejecucion_remota.sh
- procesamiento.py

5.6.1 Herramienta `simulaciones.sh`

Esta es la herramienta principal, y es un fichero escrito en ShellScript [93], de nombre simulaciones.sh que permite, a partir de un escenario inicial, realizar una tanda de NUM_SIM simulaciones (por ejemplo $NUM\_SIM = 30$) donde lo único que cambia entre una simulación y la siguiente es la semilla para la simulación y la posición de las motas. Una vez realizada la tanda de NUM_SIM simulaciones, se habrán obtenido un conjunto de NUM_SIM ficheros de resultados, que contienen la información necesaria para generar las métricas especificadas en la Tabla 5.1 para cada simulación. Para cada tanda de simulaciones se habrá fijado previamente:

- $TIPO\_RPL$ = esquema (un valor de *RPL*, *xRPL*, *MxRPL*, *MxRPL-AP*)
- TAM_RED = longitud del lado del cuadrado (en m)

Además también se usan otros parámetros:

- $NUM\_SIM = 30$ (número de simulaciones)
- $DURACION = 36000000$ (tiempo total de la simulación en *ms*, equivalente a 10 *horas*)
- $PERIODO = 10$ (tiempo que espera cada mota para enviar el siguiente paquete de datos en s)

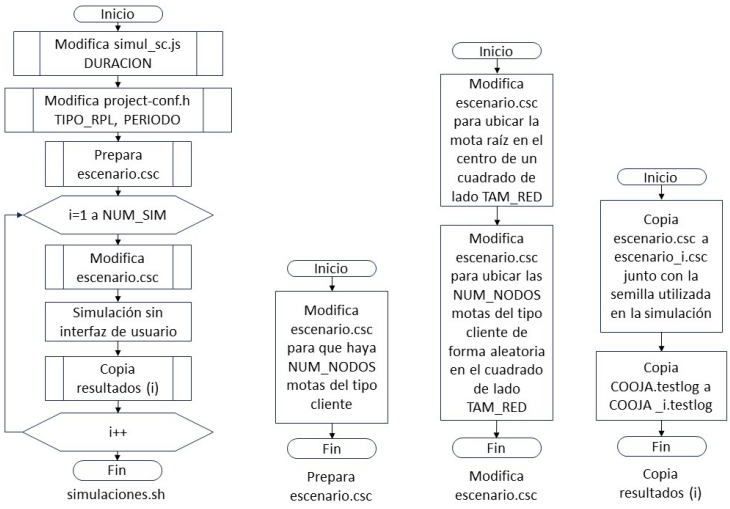


Figura 5.3 Funcionamiento de la herramienta simulaciones.sh.

- $NUM\_NODOS = 15$ (número de motas clientes, no se cuenta la mota raíz)

Antes de empezar con las simulaciones automatizadas, se debe ejecutar una primera vez el simulador Cooja con la interfaz de usuario activada para crear un fichero escenario.csc con:

- una mota raíz (con el código udp−server.c)
- una mota cliente (con el código udp−client.c)
- configuración del complemento 'Simulation Script Editor' con el fichero simul_sc.js
- activación del complemento PowerTracker

El funcionamiento de la herramienta se muestra en la Figura 5.3. Cuando termina de ejecutarse, se tienen los resultados de las simulaciones en NUM_SIM ficheros de nombre COOJA_i.testlog, donde i es el número de la simulación. También se obtienen NUM_SIM ficheros de nombre escenario_i.csc, con la semilla utilizada, por si se quiere repetir alguna simulación con los mismos datos exactamente (la misma semilla y la misma ubicación de las motas). En cuanto a la semilla utilizada en cada simulación, es generada por Cooja y diferente cada vez, por eso no hay que cambiarla de forma explícita dentro del bucle, y por eso se guarda en las copias del escenario.

Con esta herramienta se puede hacer una tanda de NUM_SIM simulaciones, donde en cada simulación cambia la posición de las motas, pero no el tamaño del terreno. Para hacer simulaciones con distintos tamaños de terreno se ha desarrollado otra herramienta, que se describe a continuación.

5.6.2 Herramienta `ejecucion_remota.sh`

Esta herramienta también es un fichero escrito en ShellScript, de nombre ejecucion_remota.sh que permite usar la herramienta anterior cambiando los parámetros, especialmente el

tamaño del terreno (*TAM_RED*). De esta forma, en ejecucion_remota.sh se configura una tabla *TAMS* con los tamaños de terreno que se quieren simular, y ejecuta la herramienta simulaciones.sh con cada uno de los tamaños. También se configura un esquema *TIPO_RPL*, la duración de la simulación *DURACION*, el periodo *PERIODO*, y el número de motas *NUM_NODOS*. A los ficheros de resultados les cambia el nombre para que además del número de la simulación, también reflejen los parámetros:

- *TAM_RED*
- *TIPO_RPL*
- *DURACION*
- *PERIODO*
- *NUM_NODOS*

Además, para aprovechar la potencia de cálculo de los servidores disponibles, cada ejecución de simulaciones.sh se puede ejecutar en máquinas virtuales en distintos servidores, que se han preparado previamente y en los que se ha instalado Cooja. La comunicación se realiza mediante scp para copiar los ficheros y ssh para enviar las órdenes. Con esta herramienta (que hace uso de la anterior), si se tiene una tabla *TAMS* con 8 tamaños de terreno, y se ha configurado *NUM_SIM* para realizar 30 simulaciones con el mismo tamaño de terreno, entonces se realizan 240 simulaciones:

- $TAMS = (010\ 020\ 030\ 040\ 050\ 060\ 080\ 100)$
- $NUM\_SIM = 30$
- $long(TAMS) \times NUM\_SIM = 8 \times 30 = 240$

Todas estas simulaciones serían del mismo esquema *TIPO_RPL*, pero con distinto tamaño de terreno *TAM_RED*. Para hacer las simulaciones para otro esquema, sólo hay que cambiar *TIPO_RPL* y volver a ejecutar la herramienta.

El resultado de estas simulaciones son copias del fichero COOJA.testlog, que contienen la información que se especificó en el apartado 5.3.4, para cada mota, pero esta información hay que procesarla para obtener las métricas especificadas en la Tabla 5.1, y hay que hacer las medias correspondientes. Esto se hace con la herramienta que se describe a continuación.

5.6.3 Herramienta `procesamiento.py`

Esta herramienta está basada en ficheros que contienen scripts escritos en Python, que procesa los ficheros generados en cada simulación, en varias fases.

En la primera fase se procesa el fichero de cada simulación y se obtiene un fichero csv, con los valores de las métricas para cada mota, y el valor medio de cada métrica para todas las motas de tipo cliente (no se tiene en cuenta el raíz, que es servidor). Ese valor medio de cada métrica de las motas cliente es el que se usa en la siguiente fase.

En la segunda fase, se agrupan los ficheros correspondientes a una tanda de simulaciones, identificados porque pertenecen a un mismo esquema (*TIPO_RPL*) y a un mismo tamaño del terreno (*TAM_RED*), y se calcula la media de las métricas y su intervalo de confianza

de 95 %. Por ejemplo, si se procesan los $NUM\_SIM = 30$ ficheros correspondientes a una tanda de simulaciones con $TIPO\_RPL = MxRPL$ y $TAM\_RED = 080$, se obtendría un valor medio para cada métrica (y su intervalo de confianza) para ese esquema y ese tamaño de terreno. Por ejemplo, para la métrica $Retardo(ms)$ se tendría un valor para el esquema $MxRPL$ con el tamaño de terreno $80m \times 80m$.

Una vez realizada la operación para todos los esquemas y para todos los tamaños de terreno, ya se tienen los datos para generar una tabla detallada de resultados numéricos, con valores medios de cada una de las métricas de la Tabla 5.1, para cada esquema y tamaño de terreno. Esa tabla detallada se mostrará en el capítulo siguiente, junto con las gráficas que se generan a partir de ella.

6 Resultados y discusión

En este capítulo se van a estudiar los resultados de las simulaciones para responder a las cuestiones planteadas en el apartado 5.1, y analizar cómo afecta la densidad de la distribución de las motas a los distintos esquemas, valorando el consumo de energía, calidad de servicio, rendimiento de RPL e influencia del Probing alternativo.

En las simulaciones se han distribuido las 15 motas cliente en un área cuadrada con diferentes tamaños de terreno, desde $100m^2$ hasta $10000m^2$ (ver apartado 5.5.2). Para cada tamaño, se han simulado cuatro esquemas RPL diferentes (ver apartado 5.5.1):

1. RPL: predeterminado con todos los nodos operando al nivel de potencia más alto.
2. xRPL: con dos niveles de potencia, niveles de potencia más alto y más bajo.
3. MxRPL: con los 5 niveles de potencia de transmisión indicados en la Tabla 5.5.
4. MxRPL-AP: con el algoritmo de Probing Alternativo aplicado a MxRPL.

Se han realizado 30 simulaciones de $10h$, donde cada uno de los 15 nodos cliente transmite un paquete de datos hacia el raíz cada $10s$, para cada tamaño de terreno y cada esquema RPL; y en cada simulación se ha cambiado aleatoriamente la semilla de simulación y la distribución de los nodos (excepto el nodo raíz, que siempre se mantiene en el centro). Estas simulaciones se han desarrollado usando las herramientas descritas en el apartado 5.6, y con los datos obtenidos se ha generado la Tabla 6.1, que contiene los valores medios. En esta tabla se han omitido los valores correspondientes a algunos tamaños de terreno, para mayor claridad (de los 8 tamaños se han dejado los 4 más significativos). Con esos datos también se generan las gráficas que se mostrarán en los siguientes apartados.

Las figuras con gráficas que se refieren a un solo esquema, muestran el nombre del esquema entre paréntesis, y aquellas que comparan varios esquemas, están rotuladas con distintas líneas de puntos y colores. Los resultados mostrados por las líneas son los valores medios (Tabla 6.1), y el intervalo de confianza del 95 % se indica mediante un color sombreado.

Tabla 6.1 Resultados promedio de 30 simulaciones de 10 *h*.

	Tamaño	10m×10m				20m×20m				50m×50m				100m×100m			
	Esquema	RPL	xRPL	MxRPL	MxRPL-AP	RPL	xRPL	MxRPL	MxRPL-AP	RPL	xRPL	MxRPL	MxRPL-AP	RPL	xRPL	MxRPL	MxRPL-AP
Energía	Energía *TOTAL* (mJ)	66686,4	66382,1	66876,3	67005,5	66690,6	64197,1	64920,3	64851,7	66290,8	65315,2	63383,4	63158,8	60495,3	60080,9	59324,5	59231,3
	Energía CPU_{act} (mJ)	9048,6	9133,1	9601,8	9729,1	9055,9	8989,5	9439,4	9552,0	8841,8	8712,9	8940,2	9014,1	5734,7	5774,2	5990,5	5920,0
	Energía CPU_{LPM} (mJ)	56,6	56,6	56,5	56,5	56,6	56,6	56,6	56,5	56,6	56,6	56,6	56,6	57,0	57,0	56,9	56,9
	Energía $RADIO_{IDLE}$ (mJ)	48876,2	48876,6	48876,6	48875,7	48876,3	48924,9	48919,2	48922,5	48880,6	48898,8	48947,4	48954,0	48942,5	48952,1	48973,9	48974,4
	Energía $RADIO_{RX}$ (mJ)	8265,1	8251,3	8252,9	8288,1	8262,1	6119,1	6372,2	6226,0	8071,7	7269,6	5126,3	4832,8	5215,7	4815,0	3850,8	3824,1
	Energía $RADIO_{TX}$ (mJ)	440,0	64,5	88,5	56,0	439,7	107,0	133,0	94,6	440,1	377,3	312,9	301,3	545,3	482,5	452,4	455,9
Tráfico y QoS	APP	3595,3	3595,4	3595,2	3595,1	3595,3	3595,3	3595,2	3595,2	3595,3	3595,3	3595,3	3595,3	3587,4	3587,2	3563,5	3587,4
	APP-1	3595,3	24,1	1,35	1,28	3595,3	380,4	1,23	2,4	3595,3	3063,6	75,2	84,9	3587,4	3332,8	1652,8	1680,1
	APP-2			3,3	2,8			2,9	4,6			342,1	325,5			606,5	627,1
	APP-3			10,5	7,51			12,9	14,7			1876,4	1796,8			762,1	826,4
	APP-4			189,9	94,2			891,3	518,2			829,7	832,2			329,9	277,1
	APP-5		3571,3	3390,2	3489,3		3214,8	2686,9	3055,4		531,7	471,9	556,0		254,4	212,2	176,7
	Tramas no asentidas	0,1	0,1	0,3	0,3	0,1	0,1	0,2	0,2	0,1	0,1	0,1	0,1	0,0	0,0	0,1	0,1
	Tramas asentidas	4054,7	4053,8	4062,8	4062,0	4053,4	4052,7	4062,2	4060,9	4055,0	4062,6	4057,3	4056,5	4860,8	4723,3	4756,2	4787,2
	Retransmisiones	23,3	17,6	15,2	22,8	24,8	23,2	21,8	29,0	25,7	22,8	22,0	28,6	32,2	22,7	35,9	33,6
	Retardo (ms)	28,4	27,9	27,9	27,8	28,4	27,8	27,9	27,9	28,4	27,8	27,8	27,8	34,8	33,2	33,6	33,6
RPL	U-DIO	411,1	414,1	423,7	422,3	410,8	413,4	423,1	421,8	411,6	412,4	417,7	416,8	370,8	380,2	404,4	397,6
	U-DIO-1	411,1	271,0	134,9	4,77	410,8	312,2	138,8	8,9	411,6	379,1	200,9	93,4	370,8	362,0	232,2	167,9
	U-DIO-2			109,7	4,0			112,8	39,9			126,3	141,2			105,5	113,9
	U-DIO-3			98,4	36,8			101,7	103,5			64,2	107,8			49,4	71,9
	U-DIO-4			49,5	249,5			46,0	200,0			18,6	57,1			13,1	32,2
	U-DIO-5		143,1	31,2	127,3		101,2	23,9	69,5		33,3	7,70	17,2		18,2	4,16	11,7
	M-DIO	42,7	42,8	43,0	42,8	42,8	42,7	42,8	42,9	42,7	42,9	43,2	43,0	42,9	43,1	42,9	43,1
	M-DIO-1	42,7	22,7	10,8	10,6	42,8	22,6	10,7	10,7	42,7	22,7	11,0	10,9	42,9	22,9	10,9	10,9
	M-DIO-2			8,2	8,2			8,2	8,2			8,2	8,1			8,2	8,3
	M-DIO-3			8,0	8,0			8,0	8,0			8,0	8,0			7,9	8,0
	M-DIO-4			8,0	8,0			8,0	8,0			8,0	8,0			7,9	8,0
	M-DIO-5		20,1	8,0	8,0		20,1	8,0	8,0		20,2	8,0	8,0		20,2	7,9	7,98
	Cambios de padre	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,3	1,3	1,6	1,7
	Retardo de unión (s)	48,1	47,0	48,3	49,3	47,6	48,1	48,4	48,1	48,0	48,0	47,8	47,9	47,2	46,6	46,6	47,2

6.1 Consumo de energía

Para estudiar el consumo de energía se van a analizar las métricas que aparecen en la Tabla 6.1 en el grupo de filas correspondientes a *Energía*. Si se analiza la fila *Energía TOTAL (mJ)*, se obtiene la gráfica de la Figura 6.1, donde se representa la totalidad del gasto energético (*mJ*), es decir, la suma de los 15 nodos cliente durante las 10 *h*, correspondiente a los diferentes esquemas, frente a distintos tamaños del terreno (longitud del lado del cuadrado en *m*). En esta gráfica se observan 3 hechos:

1. Consumo decreciente: En términos generales, el consumo energético global es más alto cuando el tamaño del terreno es pequeño (y la densidad de motas es alta) y va disminuyendo con la dispersión, cuando el tamaño del terreno aumenta (motas más alejadas y menor densidad) bajando hasta un 10 %.
2. Diferencias para grandes dimensiones: Aunque para dimensiones del terreno pequeñas el consumo de todos los esquemas es prácticamente el mismo, para dimensiones grandes los consumos difieren.
3. Esquema con menor consumo: MxRPL-AP (junto con MxRPL) es el esquema que tiene menor consumo total, excepto para dimensiones de terreno pequeñas (lado del cuadrado $\leq 20m$), donde el de xRPL es menor.

Se va a razonar una explicación a esos 3 hechos en los siguientes apartados.

6.1.1 Consumo decreciente

Para explicar a qué es debido este decremento del consumo, se recuerda que, tal como se indicó en la ecuación 5.1, el consumo total se puede desglosar en dos partes, la parte

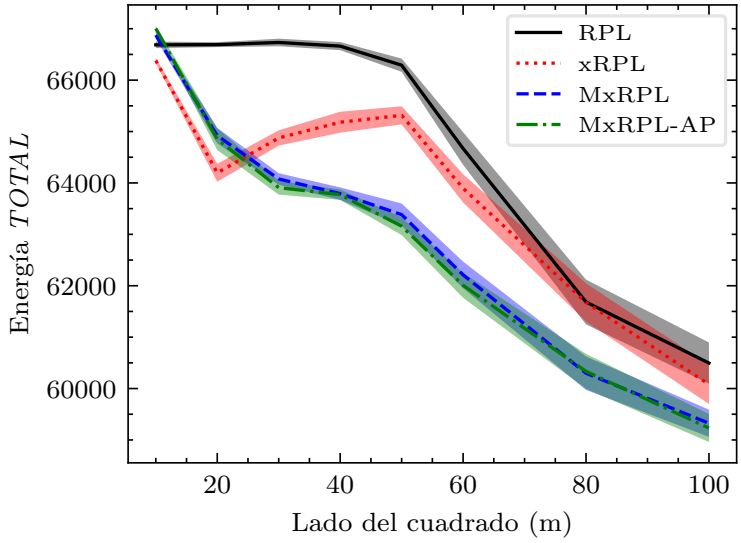
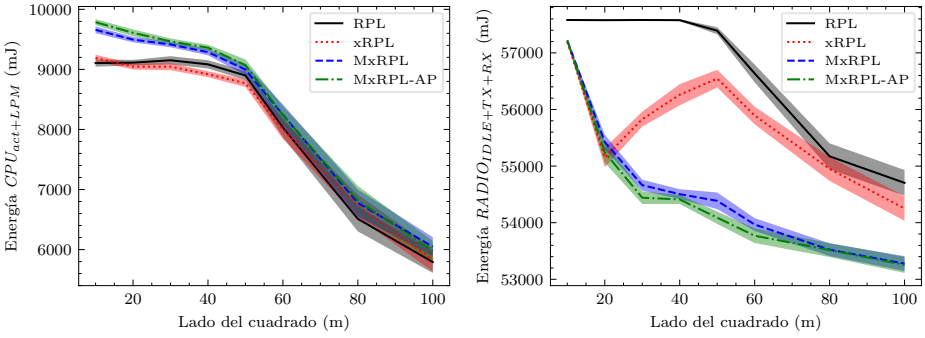


Figura 6.1 Energía consumida Total por 15 motas en 10 horas.

correspondiente a la CPU y la parte correspondiente a la radio, como muestran las dos gráficas de la Figura 6.2. En estas dos gráficas se observa que tanto el consumo de la CPU, como el de la radio, se decrementan. En el caso de la CPU, la gráfica de la Figura 6.2a refleja



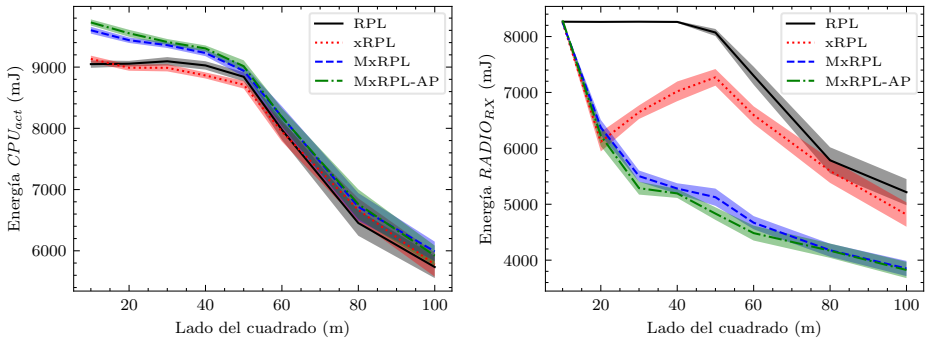
(a) CPU. **(b)** Radio.

Figura 6.2 Energía consumida desglosada por CPU y radio.

la suma de las filas *Energía CPU_{act} (mJ)* y *Energía CPU_{LPM} (mJ)* de la Tabla 6.1, y en ellas se observa que la energía consumida por la CPU en modo LPM es despreciable (del orden de 1 *mJ*) frente a la consumida en modo activo (del orden de 3000 *mJ*, aproximadamente un 30 % del consumo de la CPU), como se muestra en la Figura 6.3a.

En el caso de la radio, la gráfica de la Figura 6.2b refleja la suma de las filas *Energía*

RADIO$_{IDLE}$ (mJ), *Energía RADIO$_{TX}$ (mJ)* y *Energía RADIO$_{RX}$ (mJ)* de la Tabla 6.1. Al estudiar la variación de *Energía RADIO$_{IDLE}$ (mJ)* se observa que es del orden de 100 *mJ*, y la variación de *Energía RADIO$_{TX}$ (mJ)* es del orden de 500 *mJ*, pero en cambio, la variación en *Energía RADIO$_{RX}$ (mJ)* es del orden de 4000 *mJ* (aproximadamente un 50 % del consumo de la radio), como se muestra en la Figura 6.3b.



(a) CPU Modo activo.

(b) Radio RX.

Figura 6.3 Energía consumida por CPU en modo activo y Radio en recepción.

El menor consumo de CPU en modo activo y de la radio en recepción al aumentar la dispersión, se puede atribuir al hecho de que los nodos más alejados entre sí tienen menos vecinos, produciéndose así menos tráfico de control. El aumento de consumo de la radio en recepción en xRPL entre el lado del cuadrado 20 *m* y 50 *m*, donde se pasa de 6100 *mJ* a 7300 *mJ*, cuando en general el consumo disminuye con el aumento del lado del cuadrado, se debe a que para ese caso el número de vecinos aumenta, tal como se explica en apartado 6.2.2.

6.1.2 Diferencias para grandes dimensiones

El hecho de que para pequeñas dimensiones del terreno ($10m \times 10m$) el consumo total sea prácticamente igual para todos los esquemas se debe a que debido a la cercanía de los nodos, todos los nodos son vecinos y se escuchan entre ellos, por lo que el consumo de la CPU en modo activo y la radio en recepción (los dos aspectos que se han visto en el apartado anterior que influyen más en el consumo) es similar para todos los esquemas, tal como se mostró en la gráficas de las Figura 6.3.

Para dimensiones intermedias del terreno (hasta $50m \times 50m$) los consumos difieren, debido a que en los distintos esquemas se transmite a distintas potencias. La potencia media a la que se transmiten los paquetes de datos en los distintos esquemas viene representada en la gráfica de la Figura 6.4. Esta gráfica se calcula teniendo en cuenta las potencias de transmisión (en *mW*) que se especificaron en la Tabla 5.5 y haciendo la media ponderada con los indicadores *APP-i* (dentro del grupo *Tráfico y QoS* de la Tabla 6.1). En esta gráfica se observa que las potencias usadas en xRPL son mayores que en MxRPL y MxRPL-AP, y que además van aumentando su diferencia a medida que aumenta el tamaño del terreno.

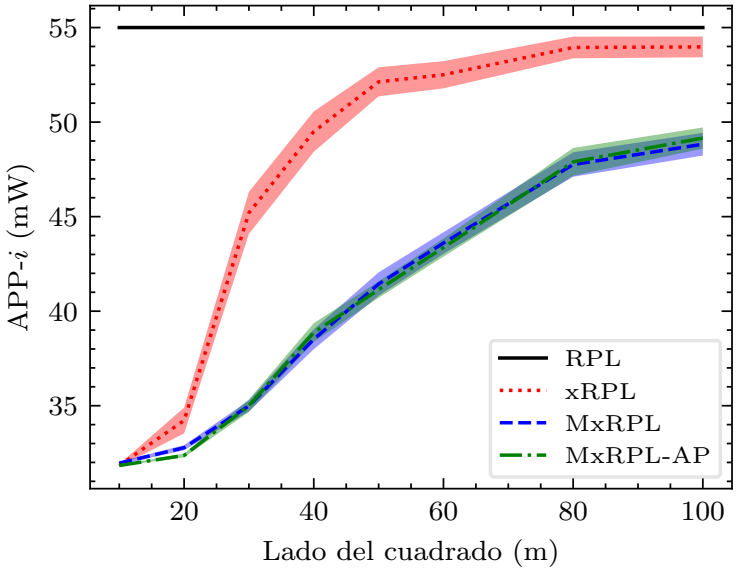


Figura 6.4 Potencia de transmisión media de paquetes de datos.

Debido a estas distintas potencias, el número de vecinos (nodos conocidos) cambia en los distintos esquemas, y eso afecta al consumo de CPU y radio en recepción.

Para mayores dimensiones del terreno (a partir de $50m \times 50m$), la diferencia entre las potencias empieza a disminuir, ya que en xRPL se está cerca de la potencia máxima y no puede aumentar más, y en cambio en MxRPL y MxRPL-AP se va aumentando la potencia paulatinamente, pero sin llegar al máximo. Este acercamiento hace que los consumos empiecen a converger, pero en las dimensiones máximas simuladas ($100m \times 100m$) todavía no son iguales, debido a que en xRPL se transmite principalmente a máxima potencia y en MxRPL-AP se transmite a potencias intermedias. Esta diferencia hace que el número de vecinos sea distinto, y el consumo de la radio en recepción también (ver gráfica de la Figura 6.3b). Si se siguiera aumentando las dimensiones, en todos los esquemas se transmitiría a máxima potencia, el número de vecinos sería similar y el consumo también, con lo cual terminarían convergiendo.

6.1.3 Esquema con menor consumo

La gráfica de la Figura 6.1 muestra que para dimensiones de terreno pequeñas, hasta lado del cuadrado de $25m$, el esquema que tiene menor consumo total de energía es xRPL, pero a partir de ahí, el esquema de menor consumo es MxRPL (y también MxRPL-AP).

En tamaños de terreno pequeños, los nodos crean una topología en estrella, estableciendo su potencia de transmisión al mínimo (se puede comprobar cómo el indicador $APP - 5$ de la Tabla 6.1 disminuye con el tamaño del terreno). MxRPL tiene un rendimiento ligeramente inferior que xRPL (aunque mejor que RPL) en estos tamaños pequeños debido a la generación de más mensajes de control (DIO unicast) que también deben ser procesados

por los nodos (la Figura 6.3a muestra que el consumo de energía de la CPU en modo activo es mayor en MxRPL que en xRPL o RPL).

Sin embargo, más allá de los $25m \times 25m$ ($41m^2/nodo$), MxRPL-AP muestra el mejor rendimiento. Esto se debe a que los nodos tienen que aumentar su potencia de transmisión para poder alcanzar a los nodos vecinos. Mientras que en xRPL algunos nodos se ven obligados a cambiar a su nivel máximo de potencia (lo que aumenta notablemente el consumo de energía en los estados de transmisión y recepción), en MxRPL los nodos cambian al siguiente nivel de potencia, sin tener que llegar al máximo, lo que resulta en una notable reducción del gasto de energía en transmisión y recepción con respecto a xRPL y RPL.

La distancia promedio entre nodos aumenta al tratar con tamaños de terreno medianos a grandes, que miden más de $50m \times 50m$. Como consecuencia, tiende a disminuir el número de vecinos, y los nodos utilizan un nivel de potencia de transmisión más alto, lo que resulta en menos escucha (es decir, menos energía en recepción) y una convergencia gradual entre todos los enfoques. De hecho, a medida que aumenta la dispersión ($80m \times 80m$ y más), todos los enfoques aumentan su potencia de transmisión y disminuyen su consumo en recepción. En este rango, MxRPL también tiene un rendimiento superior a los otros enfoques, ya que la mayoría de los nodos establecen un nivel de potencia inferior al máximo, lo que resulta en menos vecinos y menos energía de recepción en promedio.

Por lo tanto, los resultados de la Figura 6.1 nos permiten concluir que, con los niveles de potencia de transmisión utilizados en las simulaciones, MxRPL supera a xRPL y al estándar RPL en la mayoría de los tamaños de terreno estudiados. Esto se atribuye a un uso más eficiente de la potencia de transmisión en los dispositivos, que se traduce en menos tráfico de control recibido, lo que a su vez ahorra energía en el consumo de la radio, logrando una reducción de hasta un 20 % en la transmisión y un 34 % en la recepción en comparación con xRPL, y hasta un 26 % en la transmisión y un 40 % en la recepción en comparación con el RPL estándar.

6.2 Calidad de servicio

En este apartado se van a analizar las métricas correspondientes al grupo *Tráfico y QoS* de la Tabla 6.1.

6.2.1 Retardo, transmisiones y retransmisiones

Uno de los objetivos de los esquemas propuestos es no degradar la calidad de servicio, y un indicador importante de este aspecto es el retardo extremo a extremo de los paquetes de datos. Este indicador aparece en la Tabla 6.1 como $Retardo(ms)$ y se muestra en la gráfica de la Figura 6.5, donde se observa que es prácticamente igual para todos los esquemas, por lo que se puede aseverar que los esquemas propuestos no afectan a este aspecto de la calidad de servicio.

El hecho de que aumente el retardo cuando aumenta el tamaño del terreno, se debe a que al aumentar la separación entre las motas, hace falta más de un salto para que los paquetes de datos lleguen a la mota raíz. Este reenvío de paquetes de datos se puede comprobar con el aumento de transmisiones con éxito cuando aumenta el tamaño del terreno, tal

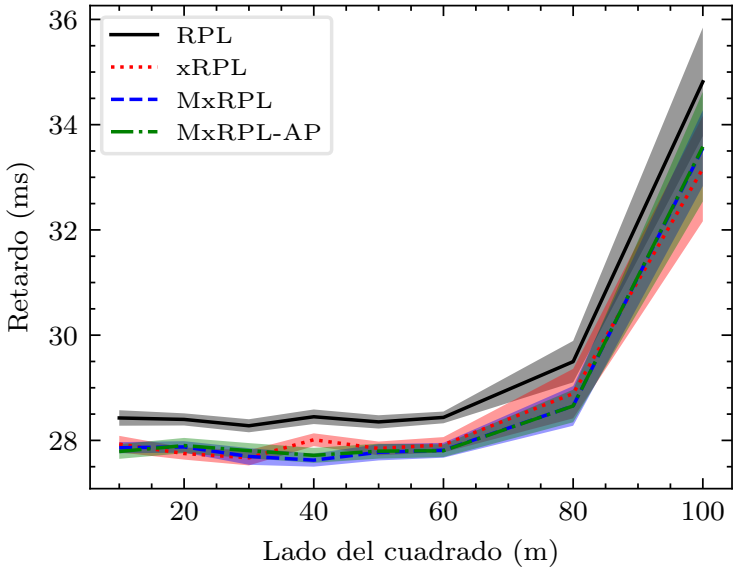


Figura 6.5 Retardo extremo a extremo.

como se puede observar en el indicador *Tramas asentidas*, y que se muestra en la gráfica de la Figura 6.6. En esta gráfica se muestra el número de tramas enviadas por cada mota cliente, de las que se recibe asentimiento (no se contabilizan las multicast). Hasta el lado de cuadrado de 60m el número de tramas enviadas coincide prácticamente con la suma de envíos de paquetes de datos y U-DIO. A partir de ahí, aunque el envío de U-DIO va disminuyendo, como se puede observar en el indicador *U-DIO*, el número de tramas recibidas aumenta, como se puede observar en el indicador *Tramas recibidas*.

Además de los indicadores mencionados, dentro del grupo *Tráfico y QoS*, también se pueden analizar *Retransmisiones* y *Tramas no asentidas*. El primero indica el número medio de veces que una mota ha tenido que retransmitir una trama unicast hasta que se recibe el asentimiento. El segundo indica el número de medio de veces que no se ha recibido asentimiento. Se recuerda que se hace la media para las 15 motas cliente y a su vez para las 30 simulaciones de 10h. Estos indicadores tienen valores muy bajos y apenas varían para todos los esquemas, por lo que los esquemas propuestos tampoco afectan a estos aspectos de la calidad de servicio.

6.2.2 Paquetes de datos

Las gráficas de la Figura 6.7 muestran la cantidad de paquetes de datos de la aplicación enviados en cada nivel de potencia de transmisión disponible (H y L para xRPL, y 1-5 para MxRPL) con los distintos tamaños de terreno. Estos datos se obtienen de las filas $APP - i$ dentro del grupo *Tráfico y QoS* de la Tabla 6.1. Claramente, la elección del nivel de potencia de transmisión depende de la dispersión y del esquema bajo consideración.

Así, en xRPL, la mayoría de los nodos hacen la transición desde el nivel mínimo hasta

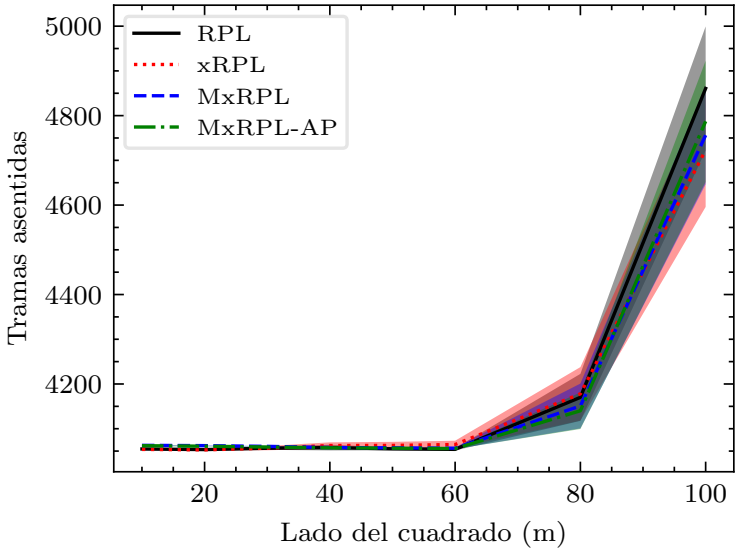
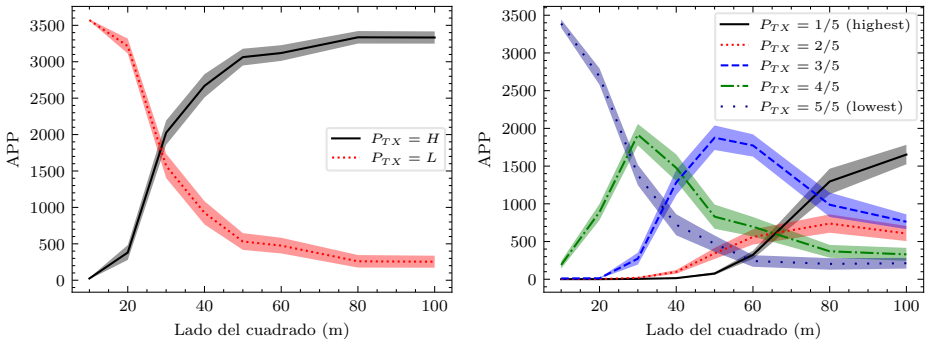


Figura 6.6 Tramas enviadas y asentidas.



(a) xRPL. **(b)** MxRPL-AP.

Figura 6.7 Transmisiones a distintas potencias de paquetes de datos.

el máximo de potencia de transmisión en escenarios entre $20m \times 20m$ y $50m \times 50m$. Este aumento tan marcado de potencia en en la transmisión de los paquetes de datos en xRPL entre el lado del cuadrado $20m$ y lado de $50m$, hace que el número de vecinos aumente, lo que explica el aumento de consumo de la radio en recepción en ese intervalo, observado en la gráfica de la Figura 6.3b, donde el consumo de la radio en recepción pasaba de $6100mJ$ en $20m$ a $7300mj$ en $50m$, rompiendo la tendencia general de disminución del consumo cuando el tamaño del terreno aumenta. En el tamaño de terreno mayor, de $100m \times 100m$ ($666m^2/nodo$), el 98 % de los paquetes de datos se transmiten en xRPL utilizando el nivel

máximo de potencia de transmisión, por lo que hay poca diferencia entre xRPL y RPL estándar, como se muestra en la Figura 6.1.

En MxRPL, sin embargo, los nodos reaccionan a la creciente dispersión realizando una transición suave al siguiente nivel de potencia (ver Figura 6.7b). Así, en el tamaño de terreno de $100m \times 100m$, solo el 47 % de los paquetes de datos se envían en MxRPL con la máxima potencia de transmisión (en comparación con el 98 % de xRPL). En un tamaño de terreno de $50m \times 50m$, el porcentaje de paquetes de datos enviados con la máxima potencia de transmisión es solo del 2 % en MxRPL, frente al 85 % en xRPL (ver detalles en la Tabla 6.1). Esto demuestra las ventajas de MxRPL, con su granularidad, sobre xRPL en una amplia gama de escenarios medianos y grandes, tal como se mostró en la Figura 6.4.

6.3 Rendimiento de RPL

En este apartado se van a analizar las métricas correspondientes al grupo *RPL* de la Tabla 6.1.

6.3.1 Formación del árbol

El tiempo que tarda en formarse el árbol DODAG es equivalente al tiempo que tarda en unirse al árbol la última mota, y el tiempo medio que tarda una mota en unirse al DODAG, conocido como *JoinDelay*, aparece en la Tabla 6.1 como el indicador *Retardo de unión (s)*. En este indicador se observa que la media es muy parecida en todos los esquemas, por lo que se puede concluir que este indicador no es afectado por los esquemas propuestos. En la gráfica de la Figura 6.8 se puede observar que aunque aparentemente hay variación, todos los valores medios se encuentran entre los $47s$ y $50s$, valores bastante similares, y pequeños respecto al tiempo de simulación de $10h$.

Relacionado con el DODAG también está el indicador *Cambios de padre*, que indica el número medio de veces que una mota cliente cambia de padre. Sus valores son muy parecidos en todos los esquemas, y muy cercanos a la unidad, lo que significa que apenas hay cambio de padre. Cuando el tamaño del terreno aumenta, los valores se incrementan un poco, siendo los incrementos un poco mayores en MxRPL y MxRPL-AP, pero los valores medios no llegan a 2. En la gráfica de la Figura 6.9 se pueden observar estos pequeños incrementos.

6.3.2 Mensajes de control

Los mensajes de control de RPL utilizados en esta tesis son M-DIO (DIO multicast) y U-DIO (DIO unicast), que se detallaron en el apartado 2.2.

Los mensajes M-DIO se envían de forma aleatoria y periódica, a las distintas potencias de forma rotatoria. El intervalo de envío se calcula según el temporizador *Trickle*, que inicialmente tiene un periodo medio de $4s$, y si la situación es estable, ese periodo va aumentando hasta un máximo de $1000s$ (estos valores son los que se han usado en las simulaciones de esta tesis, aunque son configurables). El indicador *M-DIO* en la Tabla 6.1 muestra que se envían normalmente una media de 42 mensajes durante toda la simulación, independientemente del esquema y del tamaño del terreno. En la gráfica de la Figura 6.10

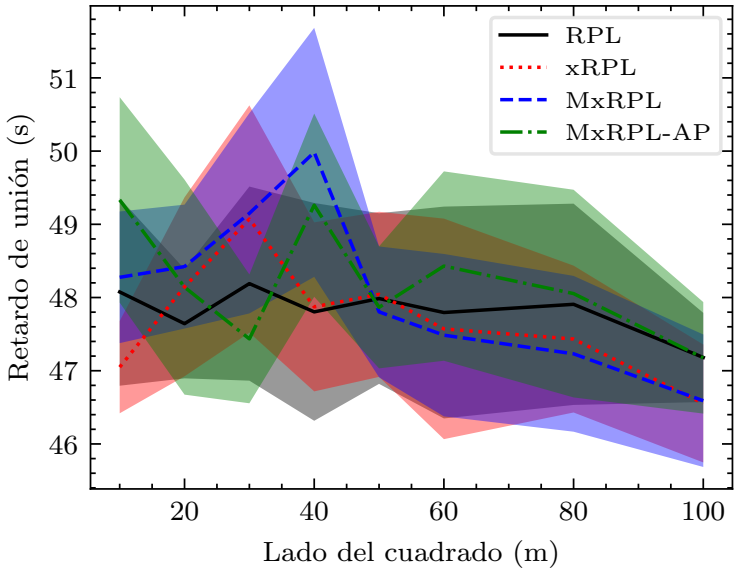


Figura 6.8 Retardo medio de una mota en unirse al DODAG.

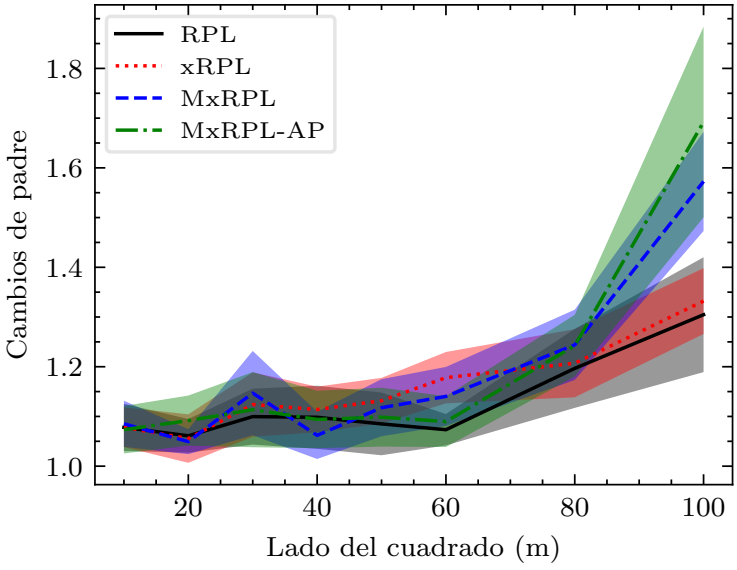


Figura 6.9 Número medio de cambios de padre.

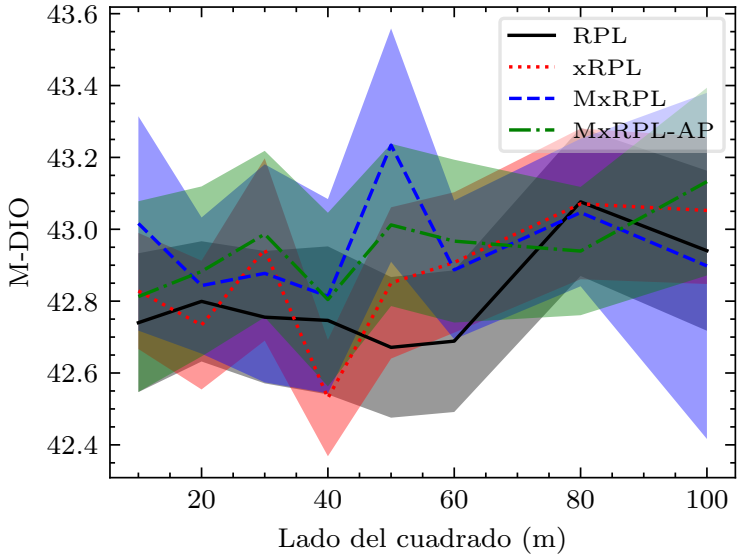


Figura 6.10 Número de mensajes M-DIO enviados por cada mota.

se puede comprobar que los valores son muy parecidos (entre 42,7 y 43,1). Los indicadores *M-DIO-i* muestran que se envían los M-DIO a todas las potencias (con un ligero incremento por la potencia máxima, que es la utilizada inicialmente cuando se une al DODAG).

Los mensajes U-DIO se envían a los nodos vecinos para el sondeo, usando los distintos algoritmos de sondeo se explicaron en el apartado 3.3. Se envían con un periodo aleatorio cuya media es 90*s* (con un valor mínimo de 45*s* y máximo de 135*s*), y el número total de mensajes enviados viene expresado en el indicador *U-DIO*, que se muestra en la gráfica de la Figura 6.11. En esta gráfica se observa que cuando aumenta el tamaño del terreno, el número de mensajes decrece, sobre todos para tamaños muy grandes. Esto se debe a que disminuye el número de vecinos, puede que no haya ningún vecino desactualizado y por lo tanto no hay que enviar mensaje de sondeo. Respecto a los distintos esquemas, con MxRPL-AP se envían más mensajes U-DIO que con xRPL (hasta un 6 %) y que con RPL (hasta un 9 %), pero se envían menos que con MxRPL (hasta un 2 %).

6.4 Influencia del algoritmo de Probing alternativo

Para estudiar la eficiencia del algoritmo de sondeo alternativo usado en MxRPL-AP, descrito en apartado 3.3.3, se va a comparar la cantidad de mensajes U-DIO relacionados con el sondeo, enviados en cada nivel de potencia de transmisión para MxRPL (sin sondeo alternativo) y MxRPL-AP (con sondeo alternativo). En la Figura 6.12 se muestra el número de mensajes U-DIO que envía cada mota a cada nivel de potencia.

Los resultados de la gráfica de la Figura 6.12a muestran que el sondeo predeterminado presenta un uso ineficiente de la energía de transmisión, predominando el uso del nivel

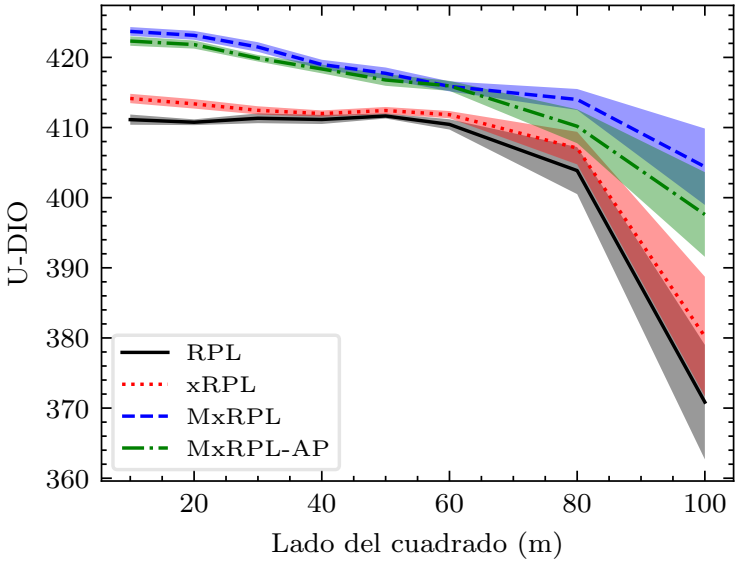
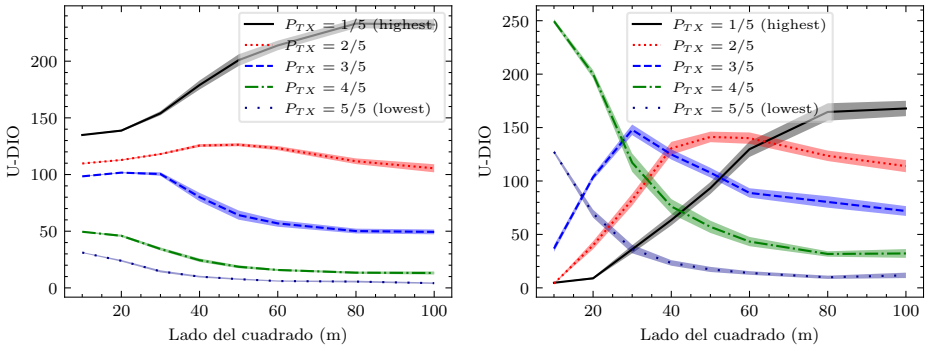


Figura 6.11 Número de mensajes U-DIO enviados por cada mota.



(a) MxRPL. **(b)** MxRPL-AP.

Figura 6.12 Transmisiones a distintas potencias de paquetes U-DIO.

de potencia máximo sobre los demás. Esto se puede atribuir a que, en general, el número de vecinos que se observan con una potencia de transmisión más alta es mayor, ya que de los vecinos que están cerca se observan todas las potencias, pero de los que están más alejados se observan sólo las potencias más altas. Teniendo en cuenta que el objetivo puede ser cualquier vecino, es más probable que se seleccione un vecino con potencia alta, y por ello se envían más U-DIO a potencia alta. Cuando el tamaño del terreno aumenta, la cantidad de vecinos que se observan con potencia baja disminuye, lo que justifica el ligero decremento de U-DIO de potencia baja cuando aumenta la dispersión.

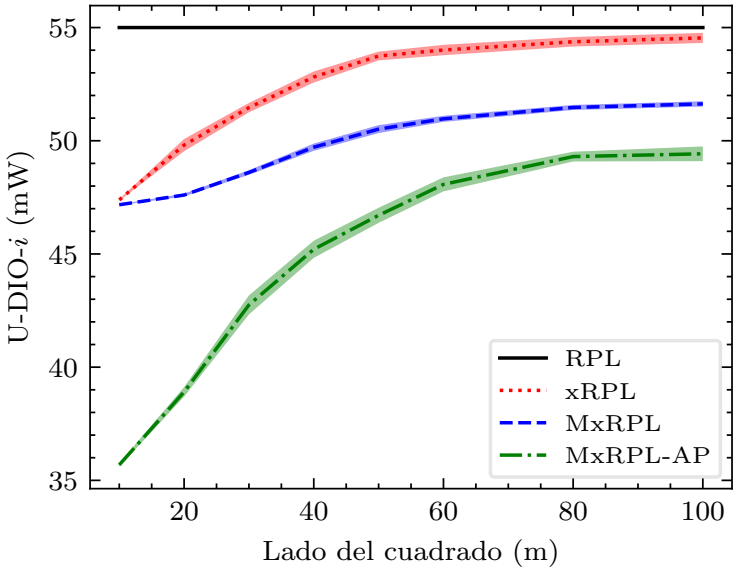


Figura 6.13 Potencia de transmisión media de U-DIO.

Sin embargo, el mecanismo de sondeo alternativo descarta los niveles de potencia altos, reduciendo así los posibles objetivos, y llegando en el peor de los casos a sólo un nivel por encima de la potencia óptima. Esto es particularmente útil en tamaños de terreno mediano y pequeño, ya que se observan todos los niveles de potencia de los vecinos, tal como se muestra en la gráfica de la Figura 6.12b.

Como resultado, el algoritmo de sondeo alternativo proporciona hasta un 24 % de ahorro de energía en transmisión en comparación con el sondeo predeterminado, como se observa en la potencia de transmisión promedio para U-DIO, mostrada en la gráfica de la Figura 6.13.

6.5 Discusión de los resultados

Según los resultados anteriores, se pueden identificar tres comportamientos diferentes según el tamaño del terreno.

Para altas densidades de nodos (menor que 26 $m^2/nodo$ ó menor que $20m \times 20m$ para 15 nodos):

- Se forma una topología de estrella, y en todos los enfoques de capa cruzada la mayoría de sus nodos transmiten al nivel de potencia más bajo.
- Debido a la proximidad, todos los nodos son vecinos, lo que resulta en un consumo de energía muy alto asociado con la recepción (sobre-escucha u "overhearing").
- Se puede concluir que xRPL es ligeramente más eficiente en consumo de energía que MxRPL gracias a su simplicidad, ya que genera, recibe y procesa menos mensajes

de control, como U-DIO. En tamaños de terreno de $20m \times 20m$, algunos nodos ligeramente más alejados comienzan a disminuir el número de vecinos, lo que se traduce en un menor consumo de energía.

Para densidades medias de nodos (entre $26m^2/nodo$ y $166m^2/nodo$ ó entre $20m \times 20m$ y $50m \times 50m$ para 15 nodos):

- Los nodos aumentan gradualmente su potencia de transmisión para alcanzar al nodo raíz a medida que aumenta la dispersión.
- En xRPL, los nodos deben cambiar al nivel máximo de potencia, ya que sólo tienen 2 niveles, lo que se traduce en un aumento de la potencia de transmisión y consumo de energía en recepción y en procesamiento, debido a la sobre-escucha. xRPL es el único enfoque que aumenta su gasto de energía en este rango de tamaño.
- Se puede concluir que en este rango, MxRPL muestra plenamente su ventaja sobre enfoques alternativos, ya que los nodos aumentan gradualmente la potencia de transmisión al próximo nivel disponible, que no es el máximo, mostrando un aumento de potencia de transmisión más gradual y reduciendo, gracias a la mayor dispersión, el número de vecinos y la energía asociada con eventos de recepción.

Para densidades bajas de nodos (entre $166m^2/nodo$ y $200m^2/nodo$ ó entre $50m \times 50m$ y $100m \times 100m$ para 15 nodos):

- En xRPL, la mayoría de los nodos ya están transmitiendo al máximo nivel de potencia de transmisión (los incrementos son modestos), por lo que el aumento de la dispersión se traduce en menos vecinos y menos energía de recepción y procesamiento.
- En MxRPL, los nodos siguen aumentando gradualmente su potencia de transmisión, pero esto no se traduce en más vecinos, debido a la mayor dispersión. Por lo tanto, MxRPL sigue mostrando una ventaja significativa en eficiencia energética sobre xRPL y RPL. Incluso en el área más grande ($100m \times 100m$), y debido a la ubicación aleatoria de los nodos (excepto el raíz), una parte significativa de los nodos elige una potencia de transmisión inferior a la máxima.
- Se puede concluir que en tamaños de terreno muy grandes y dispersos, todos los enfoques tienden a converger, ya que todos los nodos transmitirán al máximo de potencia, y el número de vecinos es similar.

6.6 Limitaciones a la generalización de los resultados

Los resultados de esta tesis presentan ciertas limitaciones que restringen la generalización de los hallazgos. Una de las limitaciones radica en el escenario y el hardware simulados, como el tamaño y el número de nodos, así como los valores de potencia de transmisión utilizados. Los resultados obtenidos son válidos teniendo en cuenta el consumo de la mota simulada ($Z1$), y con los 5 niveles de potencia escogidos (ver Tabla 5.5), con otra mota y otros patrones de potencia de transmisión los resultados podrían diferir. Además del

modelo de propagación de radio empleado en Cooja (UDGM), que puede generar áreas de cobertura que no sean completamente realistas, también hay que tener en cuenta que no se han utilizado mecanismos de *Duty Cycling* (como TSCH o ContikiMAC vistos en apartado 2.3.1) que podrían reducir el consumo de la radio en recepción. No obstante, se tiene confianza en que las tendencias abordadas en esta tesis probablemente se mantendrían, incluso con estas limitaciones. Otras limitaciones adicionales que han influido en los resultados son las suposiciones realizadas, como el uso de sondeo o el patrón de tráfico de la aplicación. Resultaría interesante explorar el rendimiento de MxRPL en otras aplicaciones del IoT.

La elección de limitar los niveles de potencia a cinco se hizo con el fin de equilibrar la complejidad y los beneficios de ahorro de energía. La adición de más niveles de potencia requeriría más recursos (memoria, tráfico de control, procesamiento) y ofrecería escaso beneficio en términos de ahorro de energía. No obstante, hay que indicar que, como se mencionó anteriormente, el algoritmo de sondeo podría imponer penalizaciones en nodos más distantes con tasas de actualización menos frecuentes, lo que resultaría en menos oportunidades de ser seleccionados como padre preferido.

Finalmente, se han asumido escenarios con nodos estáticos, y en caso de que los nodos mostraran un patrón de movilidad, el esquema propuesto podría introducir problemas de convergencia que requerirían un estudio adicional. En este sentido, resultaría interesante investigar más a fondo dicha situación.

7 Conclusiones y líneas de avance

Se ha desarrollado un nuevo esquema denominado MxRPL (algoritmo de capa cruzada en RPL con múltiples niveles de potencia), que permite reducir el consumo de energía en motas IEEE 802.15.4 cuando éstas disponen de varios niveles de de potencia de transmisión. Además, se ha creado un nuevo algoritmo de sondeo alternativo que limita la cantidad de mensajes de control y previene la sobrecarga. El uso de este algoritmo de sondeo con MxRPL se ha denominado MxRPL-AP.

El estudio realizado en esta tesis indica que MxRPL-AP mejora en general la eficiencia energética en comparación con xRPL o RPL estándar, manteniendo al mismo tiempo la calidad de servicio (QoS).

Sin embargo, el beneficio de MxRPL-AP depende finalmente de la disposición del despliegue, especialmente del tamaño del terreno y la densidad de los nodos. MxRPL-AP mostró un menor consumo de energía en la transmisión y recepción de radio que sus competidores en la mayoría de los tamaños de terreno probados, reduciendo el consumo de energía en la recepción y transmisión de radio hasta un 34 % y un 20 %, respectivamente. No obstante, en tamaños de terreno muy pequeños y densos (por ejemplo, inferiores a $20m \times 20m$), xRPL puede ser más ventajoso que MxRPL debido a su simplicidad (hasta un 1 % menos de consumo de energía en un área de $10m \times 10m$). El algoritmo de sondeo alternativo sugerido también reduce la cantidad de tráfico de control enviado y el consumo de energía asociado.

Por cada algoritmo propuesto, podemos concluir:

- MxRPL-AP mejora de manera significativa el consumo energético global, con la excepción de tamaños de terreno muy pequeños y densos, donde xRPL muestra un rendimiento ligeramente superior debido, principalmente, a su simplicidad (es decir, menor tiempo de CPU, función de sondeo más simple según se describe en el apartado 3.3.3) y menor cantidad de entradas en la tabla de vecinos debido a un menor número de niveles de potencia (2 frente 5).
- MxRPL logra una reducción de hasta un 20 % en la energía consumida durante la transmisión de radio en comparación con xRPL. Sin embargo, la disminución más significativa de la energía se debe a una menor energía consumida en eventos de

recepción (hasta un 34 % en comparación con xRPL), principalmente debido a una menor sobre-escucha.

- El mecanismo de sondeo alternativo (usado en MxRPL-AP) genera ahorros energéticos al reducir la energía consumida en mensajes de control (U-DIO) en comparación con el algoritmo de sondeo estándar. De este modo, la energía utilizada en la transmisión de U-DIO se reduce hasta un 24 % en los tamaños de terreno más densos. Este aspecto resulta particularmente beneficioso en tamaños de terreno pequeños y densos, donde la sobre-escucha es común, o cuando los mensajes U-DIO constituyen una proporción significativa del tráfico general debido a las bajas tasas de datos de la aplicación.

7.1 Implicaciones prácticas y limitaciones de la propuesta

Una de las implicaciones prácticas de utilizar MxRPL en despliegues del IoT de tamaño y densidad adecuados es un menor consumo de batería en los nodos que gestionan la mayor parte del tráfico. Esto se traduce directamente en la reducción del costo operativo (ya que cambiar la batería de un nodo puede resultar costoso) del despliegue del IoT.

Por otro lado, también implica que los nodos deben ejecutar software que pueda personalizarse para implementar MxRPL (por ejemplo, Contiki), lo cual constituye una limitación en los despliegues existentes. No obstante, los esquemas propuestos son compatibles con el estándar RPL, por lo que ambos esquemas podrían interoperar en un mismo despliegue.

Hay que tener en cuenta que una de las limitaciones prácticas de la propuesta es su uso en redes muy poco densas, donde los resultados serían similares a RPL. Otra limitación es que las motas puedan transmitir a un número suficiente de distintos niveles potencia, para así poder aprovechar la granularidad del esquema propuesto. También cabe señalar las limitaciones en el caso de movilidad de los nodos, donde el tráfico de control dedicado al probing pudiera tener un coste energético significativo.

7.2 Líneas de avance

El trabajo futuro incluye el estudio de MxRPL en movilidad, donde los retardos en el cambio de padre y la convergencia son esenciales. Habría que definir distintos patrones de movilidad, teniendo en cuenta las posibles trayectorias y sobre todo ver cómo afecta a la estabilidad del DODAG.

Otro aspecto sería explorar el uso de mayor número de niveles de energía, parametrizando las baterías de simulaciones en función del número total de niveles de energía utilizados.

Sería interesante también el extender la función objetivo propuesta para que tuviera en cuenta, además de la energía necesaria para alcanzar el nodo raíz, la energía residual de los nodos (i.e. la energía que queda en los nodos de la red).

Otra posibilidad es estudiar diferentes modelos de propagación de radio, distinto a UDGM de pérdida con la distancia, que produzcan topologías más diversas. Se podrían usar otros modelos como Directed Graph Radio Medium (DGRM) o Multipath Ray-tracer Medium (MRM). Incluso podría plantearse probarlo en escenarios reales, usando motas

adquiridas en el mercado actual, e incluyendo mediciones en el laboratorio. Para ello habría que encontrar aquellas motas que permitieran la modificación de su software para adaptarlas a los esquemas propuestos. Utilizando motas físicas, en vez de simulaciones, se podría verificar la validez de los modelos usados en esta tesis. Relacionado con la radio, también se puede probar la aplicación de mecanismos *Duty Cycling*, como TSCH o ContikiMAC y estudiar los efectos del apagado de la radio en la calidad de servicio.

Finalmente también se podría intentar usar otras motas que permitan transmitir a múltiples niveles de potencia, y que tengan los recursos suficientes (memoria y CPU) para probar los esquemas propuestos. Al inicio de esta tesis se encontró la mota *Z*1, pero es posible que se puedan encontrar otras motas que cumplan los requisitos, y se les pueda adaptar el software desarrollado.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] IBM, "Internet of Things (IoT)," What is internet of things? [Online]. Available: https://www.ibm.com/topics/internet-of-things#

[2] F. Righetti, C. Vallati, and G. Anastasi, "IoT applications in smart cities: A perspective into social and ethical issues," in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2018, pp. 387–392.

[3] M. R. M. Kassim, "Iot applications in smart agriculture: Issues and challenges," in *2020 IEEE conference on open systems (ICOS)*. IEEE, 2020, pp. 19–24.

[4] C. P. Kruger and G. P. Hancke, "Implementing the Internet of Things vision in industrial wireless sensor networks," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 627–632.

[5] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Requests for Comments, RFC Editor, RFC 6550, March 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6550.txt

[6] H. Lamaazi and N. Benamar, "A comprehensive survey on enhancements and limitations of the RPL protocol: A focus on the objective function," *Ad Hoc Networks*, vol. 96, p. 102001, 2020.

[7] G. Pasolini, C. Buratti, L. Feltrin, F. Zabini, C. De Castro, R. Verdone, and O. Andrisano, "Smart City Pilot Projects Using LoRa and IEEE802.15.4 Technologies," *Sensors*, vol. 18, no. 4, pp. 1–17, 2018.

[8] IEEE Computer Society. LAN/MAN Standards Committee., *IEEE Standard for Low-Rate Wireless Networks*. IEEE, 2020, vol. 2020.

[9] A. Kliem and O. Kao, "The Internet of Things Resource Management Challenge," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, 2015, pp. 483–490.

[10] R. Morabito and J. Jiménez, "IETF protocol suite for the Internet of Things: Overview and Recent Advancements," *IEEE Communications Standards Magazine*, vol. 4, no. 2, pp. 41–49, 2020.

[11] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7252

[12] T. Berners-Lee, R. T. Fielding, and L. M. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986, Jan. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc3986

[13] M. Martí, C. Garcia-Rubio, and C. Campo, "Performance evaluation of CoAP and MQTT_SN in an IoT environment," *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 31, no. 1, p. 49, 2019.

[14] OASIS, "Advanced Message Queuing Protocol (AMQP) Version 1.0," 2012. [Online]. Available: http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf

[15] R. T. Fielding, M. Nottingham, and J. Reschke, "HTTP/1.1," RFC 9112, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9112

[16] J. Postel, "User Datagram Protocol," RFC 768, Aug. 1980. [Online]. Available: https://www.rfc-editor.org/info/rfc768

[17] W. Eddy, "Transmission Control Protocol (TCP)," RFC 9293, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9293

[18] S. E. Deering and B. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 8200, Jul. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8200

[19] DARPA, "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[20] P. Thubert and J. Hui, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282, Sep. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6282

[21] A. Sikora and V. Groza, "Coexistence of IEEE802.15.4 with other Systems in the 2.4 GHz-ISM-Band," in *2005 IEEE Instrumentationand Measurement Technology Conference Proceedings*, vol. 3, 2005, pp. 1786–1791.

[22] Villa Crespo, E. y Morales Alonso, I., *Ciberseguridad IoT y su aplicación en Ciudades inteligentes*. RA-MA Editorial, 2017.

[23] Connectivity Standards Alliance, "ZigBee." [Online]. Available: https://csa-iot.org/es/todas-las-soluciones/ZigBee/

[24] Wi-SUN Alliance, "wi-sun." [Online]. Available: https://wi-sun.org/fan/

[25] O. Gaddour and A. Koubâa, "RPL in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163–3178, 2012.

[26] S. Manvi, K. R. Shobha, and S. Vastrad, "Performance Analysis of Routing Protocol for Low Power and Lossy Networks (RPL) for IoT Environment," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13776 LNCS, pp. 341–348, 2023.

[27] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," Internet Requests for Comments, RFC Editor, RFC 6552, March 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6552.txt

[28] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks," Internet Requests for Comments, RFC Editor, RFC 6551, March 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6551.txt

[29] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," Internet Requests for Comments, RFC Editor, RFC 6719, September 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6719.txt

[30] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," Internet Requests for Comments, RFC Editor, RFC 6206, March 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6206.txt

[31] A. S. Charles, K. Palanisamy, M. Venugopal, and S. Shanmugam, "RPL enhancement with E-Sigma routing method," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 6, pp. 7813–7826, 2023.

[32] M. Zhao, A. Kumar, P. H. Joo Chong, and R. Lu, "A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities," *Peer-to-Peer Networking and Applications*, vol. 10, no. 5, pp. 1232–1256, 2017.

[33] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. E. Culler, "ContikiRPL and TinyRPL: Happy Together," in *Workshop on Extending the Internet to Low Power and Lossy Networks*, 2011.

[34] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating systems for low-end devices in the internet of things: a survey," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 720–734, 2015.

[35] H.-S. Kim, J. Ko, D. E. Culler, and J. Paek, "Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2502–2525, 2017.

[36] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, 2022.

[37] Contiki-NG maintainers and contributors, “More about Contiki-NG,” 2022. [Online]. Available: https://docs.contiki-ng.org/en/develop/doc/project/More-about-Contiki-NG.html

[38] Stanford Information Networks Group, “TinyOS Documentation Wiki.” [Online]. Available: http://tinyos.stanford.edu/tinyos-wiki

[39] TinyOS, “GitHub - tinyos/nesc: Master nesc repository.” [Online]. Available: https://github.com/tinyos/nesc

[40] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, “Evaluating the Performance of RPL and 6LoWPAN in TinyOS,” *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+ SN)*, pp. 85–90, 2011.

[41] TinyOS, “GitHub - tinyprod/prod: TinyOS (less academic, more industrial orientation, rD, less filling).” [Online]. Available: https://github.com/tinyprod/prod

[42] E. Decker, “GitHub - tp-freeforall/prod: TinyOS (less academic, more industrial, rD, less filling), still a floor wax.” [Online]. Available: https://github.com/tp-freeforall/prod

[43] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34.

[44] L. Bartolozzi, T. Pecorella, and R. Fantacci, “ns-3 RPL module: IPv6 Routing Protocol for Low power and Lossy Networks,” in *Proceedings of the Fifth International Conference on Simulation Tools and Techniques*. ACM, 2012.

[45] nsnam, “ns-3 | a discrete-event network simulator for internet systems.” [Online]. Available: https://www.nsnam.org/

[46] nsnam, “Wireless Sensor Networks (i.e., 802.15.4 + 6LoWPAN + RPL).” [Online]. Available: https://www.nsnam.org/wiki/Current_Development#Wireless_Sensor_Networks_.28i.e..2C_802.15.4_.2B_6LoWPAN_.2B_RPL.29

[47] nsnam, “IoT and WSN devices: RPL implementation.” [Online]. Available: https://www.nsnam.org/wiki/GSOC2019Projects#IoT_and_WSN_devices:_RPL_implementation

[48] Contiki-NG maintainers, “Running a rpl network in cooja.” [Online]. Available: https://docs.contiki-ng.org/en/develop/doc/tutorials/Running-a-RPL-network-in-Cooja.html

[49] Ecma International, “ECMA-262.” [Online]. Available: https://ecma-international.org/publications-and-standards/standards/ecma-262/

[50] International Organization for Standardization (ISO), “Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,” International Organization for Standardization (ISO), Tech. Rep., 1996. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/s025022_ISO_IEC_7498-3_1997(E).zip

[51] A. Moschitta and I. Neri, "Power consumption Assessment in Wireless Sensor Networks," in *ICT - Energy - Concepts Towards Zero*, G. Fagas, L. Gammaitoni, D. Paul, and G. A. Berini, Eds. Rijeka: IntechOpen, 2014, ch. 9.

[52] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. N. Albuquerque, "Survey and Taxonomy of Duty Cycling Mechanisms in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 181–194, 2014.

[53] I. Standard, I. C. Society, H. Nguyen, J. Choi, M. Kang, Z. Ghassemlooy, D. Kim, S. Lim, T. Kang, and C. Lee, "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, 2012.

[54] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," *SICS Technical Report T2011:13 , ISSN 1100-3154*, pp. 1–11, 2011.

[55] D. Passos and C. V. Albuquerque, "A joint approach to routing metrics and rate adaptation in wireless mesh networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, pp. 999–1009, 2011.

[56] D. Narayan and U. Mudenagudi, "A cross-layer framework for joint routing and rate adaptation in infrastructure Wireless Mesh Networks," *Computers & Electrical Engineering*, vol. 56, pp. 113–129, 2016.

[57] D. Narayan, M. Naravani, and S. Shinde, "Performance Evaluation of Cross-Layer Routing Metrics for Multi-radio Wireless Mesh Network," in *Advances in Computing and Network Communications*. Springer, 2021, pp. 583–593.

[58] L. Alkama and L. Bouallouche-Medjkoune, "IEEE 802.15. 4 historical revolution versions: A survey," *Computing*, vol. 103, no. 1, pp. 99–131, 2021.

[59] Y. Jang, Y. Kim, S. Park, and S. Choi, "Link adaptation strategies for IEEE 802.15. 4 WPANs: Protocol design and performance evaluation," *Journal of Communications and Networks*, vol. 21, no. 4, pp. 376–384, 2019.

[60] D. Pancaroglu and S. Sen, "Load balancing for RPL-based Internet of Things: A review," *Ad Hoc Networks*, p. 102491, 2021.

[61] A. Sebastian and S. Sivagurunathan, "A survey on load balancing schemes in RPL based internet of things," *International Journal of Scientific Research in Network Security and Communication*, vol. 6, no. 3, pp. 43–49, 2018.

[62] O. Iova, F. Theoleyre, and T. Noel, "Exploiting multiple parents in RPL to improve both the network lifetime and its stability," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 610–616.

[63] L. Zhu, R. Wang, and H. Yang, "Multi-path data distribution mechanism based on rpl for energy consumption and time delay," *Information*, vol. 8, no. 4, p. 124, 2017.

[64] S. A. Alvi, F. ul Hassan, and A. N. Mian, "On the energy efficiency and stability of RPL routing protocol," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 1927–1932.

[65] K. F. Haque, A. Abdelgawad, V. P. Yanambaka, and K. Yelamarthi, "An Energy-Efficient and Reliable RPL for IoT," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. IEEE, 2020, pp. 1–2.

[66] H. Kharrufa, H. A. Al-Kashoash, and A. H. Kemp, "RPL-based routing protocols in IoT applications: A Review," *IEEE Sensors Journal*, vol. 19, no. 15, pp. 5952–5967, 2019.

[67] B. Ghaleb, A. Y. Al-Dubai, E. Ekonomou, A. Alsarhan, Y. Nasser, L. M. Mackenzie, and A. Boukerche, "A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1607–1635, 2018.

[68] O. Iova, P. Picco, T. Istomin, and C. Kiraly, "Rpl: The routing standard for the internet of things... or is it?" *IEEE Communications Magazine*, vol. 54, no. 12, pp. 16–22, 2016.

[69] S. Tadigotla and J. K. Murthy, "A Comprehensive Study on RPL Challenges," in *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*. IEEE, 2020, pp. 1–6.

[70] D. Grover, "An Optimized RPL Protocol for Energy Efficient IoT Networks," in *2023 International Conference on Data Science and Network Security (ICDSNS)*. IEEE, 2023, pp. 01–05.

[71] H. Pereira, G. L. Moritz, R. D. Souza, A. Munaretto, and M. Fonseca, "Increased network lifetime and load balancing based on network interface average power metric for RPL," *IEEE Access*, vol. 8, pp. 48 686–48 696, 2020.

[72] S. S. Solapure and H. H. Kenchannavar, "Design and analysis of RPL objective functions using variant routing metrics for IoT applications," *Wireless Networks*, vol. 26, pp. 4637–4656, 2020.

[73] S. Capone, R. Brama, N. Accettura, D. Striccoli, and G. Boggia, "An energy efficient and reliable composite metric for RPL organized networks," in *2014 12th IEEE International Conference on Embedded and Ubiquitous Computing*. IEEE, 2014, pp. 178–184.

[74] R. B. Abdessalem and N. Tabbane, "RPL-SCSP: a network-MAC cross-layer design for wireless sensor networks," in *Proceedings of Ninth International Conference on Wireless Communication and Sensor Networks*. Springer, 2014, pp. 27–35.

[75] B. Safaei, A. M. H. Monazzah, and A. Ejlali, "ELITE: An Elaborated Cross-Layer RPL Objective Function to Achieve Energy Efficiency in Internet-of-Things Devices," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1169–1182, 2020.

[76] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, "Real-time power-aware routing in sensor networks," in *200614th IEEE International Workshop on Quality of Service*. IEEE, 2006, pp. 83–92.

[77] W. Rukpakavong, I. Phillips, L. Guan, and G. Oikonomou, "RPL router discovery for supporting energy-efficient transmission in single-hop 6LoWPAN," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 5721–5725.

[78] H.-S. Kim, J. Paek, D. E. Culler, and S. Bahk, "PC-RPL: Joint control of routing topology and transmission power in real low-power and lossy networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 2, pp. 1–32, 2020.

[79] A. Qolami, M. Nassiri, and H. Abdoli, "A transmission power control mechanism for 802.15. 4+ rpl-operated wireless sensor network," *International Journal of Sensors Wireless Communications and Control*, vol. 10, no. 2, pp. 197–206, 2020.

[80] R. Estepa, A. Estepa, G. Madinabeitia, and E. García, "RPL Cross-Layer Scheme for IEEE 802.15. 4 IoT Devices With Adjustable Transmit Power," *IEEE Access*, vol. 9, pp. 120 689–120 703, 2021.

[81] K. A. Darabkh, M. Al-Akhras, J. N. Zomot, and M. Atiquzzaman, "RPL routing protocol over IoT: A comprehensive survey, recent advances, insights, bibliometric analysis, recommendations, and future directions," *Journal of Network and Computer Applications*, p. 103476, 2022.

[82] W. A. Simpson, D. T. Narten, E. Nordmark, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," RFC 4861, Sep. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4861

[83] A. Parasuram, D. Culler, and R. Katz, "An analysis of the RPL routing standard for low power and lossy networks," Electrical Engineering and Computer Sciences, University of California at Berkeley, Tech. Rep., 2016.

[84] Texas Instruments, "CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver." [Online]. Available: https://www.ti.com/lit/ds/symlink/cc2420.pdf

[85] J. Ternero, "Implementación de MxRPL y MxRPL-AP." [Online]. Available: https://gitlab.com/jternero/contiki-ng-mxrpl-ap

[86] Texas Instruments, "MSP430F261x, MSP430F241x Mixed-Signal Microcontrollers." [Online]. Available: https://www.ti.com/lit/ds/symlink/msp430f2617.pdf

[87] K. Roussel, O. Zendra *et al.*, "Using Cooja for WSN simulations: Some new uses and limits," in *EWSN 2016—NextMote workshop*. Junction Publishing, 2016, pp. 319–324.

[88] A. Velinov and A. Mileva, "Running and testing applications for Contiki OS using Cooja simulator," *International Conference on Information Technology and Development of Education – ITRO 2016*, pp. 279–285, 2016.

[89] A. Behal, J. K. Sandhu, and G. Gupta, "Using The Cooja Simulator, Analysing The Routing Protocol (RPL) For Low Power And Lossy Networks In IoT," in *2023 IEEE International Students'Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE, 2023, pp. 1–4.

[90] A. Mahmud, F. Hossain, T. A. Choity, and F. Juhin, "Simulation and comparison of RPL, 6Lowpan, and Coap protocols using Cooja simulator," in *Proceedings of International Joint Conference on Computational Intelligence: IJCCI 2018*. Springer, 2020, pp. 317–326.

[91] Contiki-NG maintainers, "Energest – Contiki-NG documentation." [Online]. Available: https://docs.contiki-ng.org/en/develop/doc/programming/Energest.html

[92] I. N. R. Hendrawan and I. G. N. W. Arsa, "Zolertia Z1 energy usage simulation with Cooja simulator," in *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*. IEEE, 2017, pp. 147–152.

[93] The IEEE and The Open Group, "Shell Command Language." [Online]. Available: https://pubs.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html