Evaluación de tecnologías inalámbricas de área personal

Tesis Doctoral

José Manuel Cano García

Escuela Técnica Superior de Ingeniería de Telecomunicación

Málaga, Enero de 2016

Dña. Eva González Parada García, Profesora Titular del Departamento de Tecnología Electrónica de la Universidad de Málaga, y D. Juan Carlos Cano Escribá, Catedrático del Departamento de Informática de Sistemas y Computadores de la Universidad Politécnica de Valencia

Informan:

Que D. José Manuel Cano García, Ingeniero de Telecomunicación, ha realizado en el Departamento de Tecnología Electrónica de la Universidad de Málaga, bajo su dirección el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

Evaluación de tecnologías inalámbricas de área personal

Revisado el presente trabajo, estiman que puede ser presentado al Tribunal que ha de juzgarlo, por lo que AUTORIZAN y AVALAN la presentación de esta Tesis en la Universidad de Málaga.

Málaga, a 11 de Noviembre de 2015

Fdo. Eva González Parada
Profesora Titular del Dpto. de Tecnología Electrónica de la Universidad de Málaga

Fdo. Juan Carlos Cano Escribá
Catedrático del Dpto. de Informática de Sistemas y Computadores de la Universidad Politécnica de Valencia

Departamento de Tecnología Electrónica
E.T.S.I. Telecomunicación
Universidad de Málaga

Tesis Doctoral

Evaluación de tecnologías inalámbricas de área personal

Autor:
José Manuel Cano García
Ingeniero de Telecomunicación

Directores:
Eva González Parada
Doctora Ingeniera de Telecomunicación

Juan Carlos Cano Escribá
Doctor Ingeniero en Informática

Tú no puedes volver atrás
porque la vida ya te empuja
como un aullido interminable.

Te sentirás acorralada
te sentirás perdida o sola
tal vez querrás no haber nacido.

Entonces siempre acuérdate
de lo que un día yo escribí
pensando en ti
como ahora pienso.

Nunca te entregues ni te apartes
junto al camino, nunca digas
no puedo más y aquí me quedo.

Otros esperan que resistas,
que les ayude tu alegría,
que les ayude tu canción
entre sus canciones.

La vida es bella, ya verás
como a pesar de los pesares
tendrás amigos, tendrás amor,
tendrás amigos.

(Extracto de *Palabras para Julia* por *José Agustín Goytisolo*)

A mi familia y amigos,
por estar siempre ahí.

Agradecimientos

Han sido numerosas las personas que con su apoyo han contribuido directa o indirectamente en la realización de esta Tesis. Desde estas líneas quisiera aprovechar la ocasión para manifestarles mi gratitud a todas ellas.

En primer lugar, a mi amiga y tutora Eva, a quien nunca podré agradecer lo suficiente su amistad incondicional, y el incesante aliento que me ha brindado en todo momento. Nunca podré encontrar suficientes palabras que expliquen lo que su apoyo durante todos estos años ha significado para mí. Tanto a ella como a mi tutor Juan Carlos tengo también que agradecerles su enorme paciencia conmigo y su constante labor de orientación, consejo y supervisión, así como haberme animado en todo momento a finalizar este trabajo. Muchas gracias por todo el tiempo y energías invertidos en mí.

La presentación de esta Tesis tampoco habría sido posible sin las contribuciones realizadas en colaboración con Eduardo Casilari, a quien agradezco enormemente su inestimable ayuda y la generosidad que siempre me ha mostrado. Por si esto fuera poco, también tengo que agradecerle (al igual que el resto de estudiantes de tercer ciclo de la ETSIT) su asesoramiento sobre los aspectos administrativos del Doctorado, tarea ingrata aunque necesaria que lleva con infinita paciencia y dedicación.

A Pablo Martín González y a la empresa IHMAN, que además de honrarnos con su amistad desde hace muchos años, han compartido siempre con nosotros su experiencia y conocimientos, aportándonos una perspectiva más pragmática y menos académica de la realidad, que para nosotros ha sido de incalculable valor. Junto con ellos se inició el proyecto RIAM, que tal vez sea el germen de esta Tesis, y desde entonces hemos colaborado en diferentes proyectos en los que siempre nos han transmitido la sensatez del mundo real.

En lo personal, quiero mostrar mi agradecimiento a mi familia, y en particular a mis padres, por su apoyo incondicional en todo momento. También quiero hacer extensiva mi especial gratitud a todos aquellos que se han esforzado en animarme a continuar durante estos años, y en especial a mis queridos Paula y Ángel, que han llegado a preocuparse tanto que hasta le han pedido a los Reyes Magos en alguna ocasión que «tito Jose acabe la Tesis». Gracias por mostrarme siempre vuestro cariño.

No debo tampoco olvidarme de mis compañeros y amigos, que me han acompañado a lo largo de todos estos años y me han soportado hasta en mis peores días: Javi, Edu y Rosa, Carmen, Arcadio, Fabián, Alfonso, Luis, Nacho, Francis, Alberto, Juan Pedro,...

Por último, el desarrollo de esta Tesis ha sido posible gracias al soporte institucional proporcionado mediante diversos proyectos con financiación pública, y en particular por la Comisión Interministerial de Ciencia y Tecnología (CICYT) y el Ministerio de Economía y Competitividad a través de los proyectos TEC2009-13763-C02-01 (RAMAS) y TEC2013-42711-R (NEREIDAS), así como por la Consejería de Innovación de la Junta de Andalucía a través del proyecto de excelencia P08-4198 (AVATAR).

Resumen

La enorme evolución que se ha producido en la última década en el campo de las comunicaciones inalámbricas y de las plataformas de computación para electrónica de consumo, ha convertido el denominado *Internet de las cosas* (IoT) en una realidad tangible cuya eclosión parece inminente. En este marco, las tecnologías inalámbricas de área personal se configuran como el tejido base que va a permitir la conexión de diversos objetos inteligentes, muchos de ellos con limitaciones de consumo y capacidad de proceso, a una red única de área global. Así pues, la capacidad de las tecnologías WPAN existentes para ofrecer una respuesta adecuada a las diferentes necesidades puede condicionar el éxito de la IoT y ser determinante en el desarrollo de su potencial para el soporte de aplicaciones novedosas.

En esta Tesis se lleva a cabo un estudio y evaluación de las prestaciones en términos de consumo y retardo de varias tecnologías inalámbricas para redes de área personal. En concreto, se analizan las tecnologías estándares que en el momento actual cuentan con un amplio despliegue e implantación en dispositivos y sistemas comerciales, y son estándares abiertos cuya especificación se puede consultar y para los que hay diversas herramientas de desarrollo disponibles: *Bluetooth*, *Bluetooth low energy* y *ZigBee*. Las aportaciones de esta tesis no han estado encaminadas a una comparativa entre las tecnologías estudiadas, ya que en la situación actual cada una de ellas tiene su nicho de aplicación y en la práctica no deben considerarse como rivales.

Para cada una de ellas se realiza un análisis de las distintas versiones de las especificaciones a nivel de arquitectura de protocolos y topologías de red, así como un estudio del estado del arte. Este análisis establece la base para poder desarrollar modelos analíticos del comportamiento energético de las citadas tecnologías, lo que constituye uno de los principales objetivos de la Tesis. Estos modelos han sido validados experimentalmente mediante la caracterización de diversas implementaciones de dispositivos comerciales para las tecnologías evaluadas. Durante el desarrollo de las pruebas experimentales, se ha hecho uso de herramientas ya disponibles para las pilas de protocolos de cada una de las tecnologías, así como de otras herramientas personalizadas creadas ex profeso.

La metodología de trabajo desarrollada, basada en pruebas sobre sistemas reales, permite también comprobar hasta qué punto se aproxima su comportamiento al previsto en casos más ideales, así como determinar las principales limitaciones que pueden encontrarse en el funcionamiento de la tecnología actual. Además, constituye una de las aportaciones frente a otros estudios realizados sobre las mismas tecnologías y presentes en la literatura científica. En este sentido cabe destacar que el estudio del estado del arte llevado a cabo para las diferentes tecnologías muestra que aunque los estudios teóricos y las evaluaciones realizadas mediante simulación siguen siendo necesarios, convenientes, y ampliamente utilizados, existe un creciente interés por contrastar los resultados mediante pruebas en plataformas basadas en dispositivos comerciales en los casos en los que es posible.

La evaluación de las tecnologías *ZigBee* y *Bluetooth low energy* ha demostrado que poseen cierto potencial para ofrecer el soporte necesario para conectar objetos inteligentes

a la Internet de las cosas. No obstante, ambas tecnologías presentan todavía ciertas limitaciones que están intentando vencer mediante la incorporación de nuevas características a sus especificaciones, lo que deja abierta la puerta a nuevos estudios tal y como se plantea en las líneas futuras de esta Tesis.

Abstract

The impending mass deployment of the Internet of things (IoT) has become a reality thanks to the significant breakthroughs in wireless communications and embedded computing for consumer electronics that have arisen over the last decade. In this scenario, wireless technologies supporting personal area networks stand as the fabric that may enable the access of a miriad of heterogeneous smart objetcs (some of them with significant resources and power constraints) to a globally interconnected network. Thus, the ability of the existing WPAN technologies to cope with the different requirements and achieve the required perfomance in each scenario may be a key issue in the quick development of the full potential of the IoT and its novel applications.

In this thesis several available technologies for low power wireless personal area networks are analyzed, focusing mainly on the evaluation of the current consumption and transmission delay. The study is particulary aimed at widely deployed technologies based on open standards such as *Bluetooth*, *ZigBee/802.15.4* and *Bluetooth low energy*. The contributions of this thesis are not intended to establish a comparison between the evaluated technologies since they are mainly designed for non overlapping scenarios and use cases.

For each one of these technologies, available versions of the protocol architecture specifications and network topologies are thoroughly analysed, and the related work published by the research community is also reviewed. This provides the necessary background to derive analytical models for the evaluation of the power consumption of the devices for each technology, which is one of the main goals of the Thesis. These models have been experimentally validated by conducting experiments in real testbeds based on available commercial off-the-shelf devices and development platforms. In order to perform the tests, several freely available evaluation tools as well as custom-developed ones have been used.

Experiments performed in a real testbed can also show the difference between the ideal and the real behaviour as well as point out some important practical limits of the currently available technology, and is one of the contributions of the Thesis over previous related research works on certain aspects of the evaluated technologies. In this sense, related reseach by other authors reviewed in this thesis shows a trend of increasing interest in the use of real testbeds to corroborate or clarify, whenever possible, the results obtained by mean of simulations or mathematical analysis.

The evaluation of the performance of *ZigBee* and *Bluetooth low energy* shows that both technologies have potential to support to some extent some of the challenges posed by the IoT, but also they have some weakness which their designers are trying to overcome by continuously adding new features to the standard. This will keep unveiling new research opportunities as discussed in the future work section ot this Thesis.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

6LowPAN	IPv6 over Low power Wireless Personal Area Networks.
ACK	Acknowledgement.
ACL	Asynchronous Connection-less.
AFH	Adaptative Frequency Hopping.
AODV	Adhoc On-demand Distance Vector.
API	Application Programming Interface.
APS	Application Support Layer.
ARQ	Automatic Retransmission Request.
ASB	Active Slave Broadcast.
ATT	Attribute Transport Protocol.
AWGN	Aditive White Gaussian Noise.
BAN	Body Area Network.
BB	Baseband.
BB-ADDR	Bluetooth Address.
BER	Bit Error Rate.
BNEP	Bluetooth Network Encapsulation Profile.
BR	Basic Rate.
BTLE	Bluetooth Low Energy.
BU	Backoff Unit.
BW	Bandwidth.
CAC	Channel Access Code.
CAP	Contention Access Period.
CC	Congestion Control.
CCA	Clear Channel Assesment.
CDMA	Code Division Multiple Access.
CFP	Contention Free Period.
CoAP	Constrained Application Protol.
CPU	Central Proccessing Unit.
CRC	Cyclic Redundancy Code.
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance.
DAC	Device Access Code.
DPSK	Differential PSK.
DQPSK	Differential Quaternary PSK.
DSSS	Direct Sequence Spread Spectrum.
EDR	Enhanced Data Rate.
eSCO	Extended Syncrhonous Connection Oriented.
FEC	Forward Error Correction.
FFD	Full Function Device.
FH	Frequency Hopping.
FHSS	Frequency Hopping Spread Spectrum.
FTP	File Transfer Protocol.
GATT	General Attribute Profile.
GFSK	Gaussian Frequency Shift Keying.
HCI	Host Control Interface.
HS	High Speed.
HTTP	Hypertext Transfer Protocol.
IAC	Inquiry Access Code.
IEEE	Institute of Electrical and Electronics Engineering.
IETF	Internet Engineering Task Force.
IoT	Internet of Things.
IP	Internet Protocol.
IPv6	Internet Protocol version 6.
ISI	Intersymbol Interference.
ISM	Industrial, Scientific and Medical.
kbps	kilobits per second.
L2CAP	Logical Link Control and Adaptation Protocol.
LAN	Local Area Network.
LE	Low Energy.
LL	Link Layer.
LLID	Logical Link Identificator.
LM	Link Manager.
LQI	Link Quality Indicator.
LSB	Less Significant Bit.
M2M	Machine to Machine.
MAC	Medium Access Control.
MANET	Mobile Adhoc Network.
Mbps	Megabits per second.
MCU	Microcontroller Unit.
MQTT	Message Queue Telemetry Transport.
MSB	Most Significant Bit.

MTU Maximum Transfer Unit.

NACK Negative Acknowledgement.
NAP Network Access Point.
NAT Network Address Translation.
NWK Network Layer.

OLSR Optimized Link State Routing.
OSI Open System Interconnection.
OTT One Trip Time.

PAN Personal Area Network.
PANU PAN User.
PDF Probability Density Function.
PDU Protocol Data Unit.
PER Packet Error Rate.
PHY Physical Layer.
PSB Parked Slave Broadcast.
PSK Phase Shift Keying.

QoS Quality of Service.

RAM Random Access Memory.
RDC Radio Duty Cycling.
REST Representation State Transfer.
RF Radiofrequency.
RF4CE Radio Frequency For Consumer Electronics.
RFC Request For Comment.
RFCOMM RF Serial Cable Emulation Protocol.
RFD Reduced Function Device.
RFID Radio Frequency Identification.
ROM Read Only Memory.
RPL Ipv6 Routing Protocol for Low-power and lossy networks.
RREP Route Reply.
RREQ Route Request.
RSSI Received Signal Strength Indicator.

RTO Retransmission Timeout.

SACK Selective Acknowledgement.
SCO Syncrhonous Connection Oriented.
SDP Service Discovery Protocol.
SDU Service Data Unit.
SIG Special Interest Group.
SNR Signal to noise ratio.
SoC System on Chip.
SPI Serial Peripheral Interface.
SPP Serial Port Profile.

TCP Transport Control Protocol.
TDD Time Division Duplex.
TDM Time Division Multiplexing.
TDMA Time Division Multiple Access.
TSCH Timeslotted Channel Hopping.
TTL Time to Live.

UART Universal Asynchronous Receiver Transmitter.
UDP User datagram protocol.
USB Universal Serial Bus.

VANET Vehicular Adhoc Network.

WBAN Wireless Body Area Network.
WLAN Wireless Local Area Network.
WPAN Wireless Personal Area Network.
WSAN Wireless Sensor and Actors Network.
WSN Wireless Sensor Network.

ZC ZigBee Coordinator.
ZDO ZigBee Device Object.
ZED ZigBee End Device.
ZR ZigBee Router.

Capítulo 1

Introducción

Las últimas dos décadas han sido testigo de un desarrollo espectacular de las tecnologías inalámbricas, originado primero en el campo de la telefonía móvil celular y extendido posteriormente a la comunicación de datos entre diferentes equipos. Esta creciente evolución ha dado lugar a la aparición de multitud de nuevos tipos de redes y tecnologías orientadas a diferentes ámbitos y escenarios, permitiendo la interconexión de equipos con un rango de características muy heterogéneas.

Paralelamente, y en parte impulsada por estos avances, se ha producido un notable evolución de las tecnologías *hardware* que ha permitido la irrupción de una amplia variedad de dispositivos dotados de conectividad en red y con capacidad de interactuar con otros equipos en su entorno. La aparición de sistemas en chip (SoC), que integran en un único componente la unidad de proceso (CPU), memoria, y múltiples periféricos e interfaces, ha permitido una creciente miniaturización de los dispositivos, la reducción de sus costes de fabricación y el aumento de sus capacidades. Esta transformación ha tenido a su vez una doble vertiente. Por una parte, el progesivo aumento de la memoria integrada en los procesadores para sistemas altamente empotrados (SoC microcontroladores) ha permitido dotar a éstos de mejores capacidades de comunicación e interconexión, y de una mayor inteligencia. Por otra parte, el rápido desarrollo de SoC microprocesadores con creciente capacidad de cómputo y con un consumo razonablemente reducido, ha convertido a los terminales móviles en ordenadores de bolsillo con un sistema operativo completo propio capaz de dar soporte a diferentes aplicaciones de usuario, dotados de múltiples interfaces de comunicación y capacidades multimedia. Esta misma evolución ha permitido crear plataformas de computación empotrada de bajo coste y reducido tamaño, como Raspberry Pi [BBC13], Beaglebone [Mol14] y otras similares, que cuentan con una importante capacidad de proceso y capaces de gestionar múltiples comunicaciones simultáneamente y dar soporte a aplicaciones y servicios complejos.

Este marco de continuas mejoras tecnológicas ha permitido dotar de capacidades de comunicación en red a multitud objetos cotidianos, convirtiéndolos en objetos inteligentes o *smart objects*, habilitando nuevas formas de interaccionar con ellos, y abriendo la puerta a la creación de nuevas y diversas aplicaciones. Estas aplicaciones abarcan campos muy diferentes como pueden ser la monitorización de parámetros ambientales, la domótica y automatización del hogar, sistemas de iluminación inteligentes, telemedida de contadores

de suministro (*smart metering*), seguimiento del estado de salud (*m-Health*), dispositivos vestibles (*wereables*) para entrenamiento deportivo, inteligencia ambiental, redes de transporte y ciudades inteligentes, etc.

Con objeto de proporcionar soporte a este conglomerado de aplicaciones en diferentes entornos, han emergido diferentes paradigmas de comunicación que han sido objeto de enorme interés por parte de la comunidad investigadora: redes móviles adhoc (MANETs), redes vehiculares (VANETs), redes de sensores (y actuadores) inalámbricos (WSN), etc. Estos nuevos paradigmas se plantean como una alternativa a las tradicionales redes basadas en una infraestructura controlada por un operador, y, además de a la problemática característica de los interfaces radio como la presencia de pérdidas y limitaciones en el ancho de banda, deben en cada caso hacer frente a diferentes desafíos específicos como son la alta movilidad de los nodos en el caso de las VANETs o las restricciones de capacidad de proceso, memoria y consumo en el caso de las WSN.

Las diferentes tecnologías utilizadas para proporcionar la conectividad inalámbrica en los diferentes tipos de redes y escenarios mencionados se suelen clasificar en función de la potencia de emisión y rango de alcance de los radio enlaces (de mayor a menor) en redes de área metropolitana(WMAN), redes de área local (WLAN), redes de área personal (WPAN) y redes de área corporal (WBAN). En esta división no se establece realmente una frontera claramente definida entre uno y otro tipo de red, existiendo en la actualidad un continuo de tecnologías que compiten entre sí por cubrir las necesidades de interconexión en los diferentes rangos.

La presente tesis se centra en el estudio y evaluación de tecnologías habitualmente consideradas dentro del ámbito de las redes de área personal, en las cuales el rango de cobertura de transmisión varía entre 100 metros y distancias inferiores a 1 m, y las potencias radio emitidas no superan los 100mW. La baja potencia de emisión hace adecuado este tipo de tecnologías para escenarios como las redes de sensores, y en general para aquellas aplicaciones que requieran la presencia de dispositivos con un consumo energético restringido. Esta baja potencia y el uso moderado del espectro radio permiten además a estas redes operar en bandas libres sin licencia.

En este capítulo se recoge un estado del arte de la tecnología actual para redes de área personal y de las plataformas de desarrollo existentes, para posteriormente plantear los objetivos y estructura de la tesis.

1.1. Tecnologías y estándares para redes de área personal de bajo consumo

Tanto la comunidad investigadora como la industria han dedicado un importante esfuerzo al estudio y desarrollo de protocolos de comunicación para redes de área personal orientadas a la interconexión de dispositivos heterogéneos, que se ha visto plasmado en la especificación de varios estándares orientados a diferentes casos de uso, así como la existencia de multitud de propuestas no estandarizadas para optimizar algunos aspectos en casos concretos. La estandarización es necesaria para asegurar la interoperabilidad de los dispositivos y suele ser un requisito fundamental para asegurar el despliegue masivo de

la tecnología, a costa de perder flexibilidad para adaptarse a escenarios con características o restricciones específicas.

Dentro de esta actividad destaca la labor realizada por el comité técnico IEEE 802, que estableció el grupo de trabajo 802.15 con la finalidad de estandarizar protocolos e interfaces para redes WPAN, distinguiendo entre tres clases diferenciadas por la tasa binaria de transmisión y el consumo de batería: redes de elevada tasa para aplicaciones multimedia con requisitos de ancho de banda restrictivos (802.15.3 *High Rate WPAN*), redes de tasa media para proporcionar conectividad a dispositivos de propósito general como teléfonos móviles o PDA con provisión de calidad de servicio para aplicaciones de audio (802.15.1 *Medium Rate WPAN*), y redes concebidas para aplicaciones médicas e industriales de bajo consumo sin requisitos estrictos de tasa de transferencia (802.15.4, *Low Rate WPAN*). Los estándares definidos por estos subgrupos de trabajo se centran en la especificación de los niveles inferiores de la arquitectura de protocolos (niveles OSI 1 y 2), diseñándolos de forma que sean lo suficientemente flexibles como para poder trabajar con diferentes implementaciones de los niveles superiores.

El nivel físico y de enlace descrito por la norma 802.15.1 corresponde a los niveles inferiores de la arquitectura *Bluetooth* 1.2 definida por el *Bluetooth SIG*, que mantiene también la especificación de los niveles superiores. *Bluetooth* es una tecnología que ha sido adoptada desde hace tiempo por los terminales móviles para la formación de redes expontáneas de corto alcance y por ello en la actualidad su despliegue es masivo. Desde la irrupción de esta tecnología, el *Bluetooth SIG* ha ido añadiendo nuevas especificaciones de los niveles 1 y 2 para mejorar la velocidad y el consumo, que han derivado en las variantes *Bluetooth High Speed* y *Bluetooth Low Energy*. Esta última está empezando a ser adoptada de forma multitudinaria tanto por terminales móviles como por diferentes tipos de dispositivos periféricos con objeto de aumentar su autonomía. Estas tecnologías se describen con detalle en los capítulos 2 y 4.

802.15.4 es el estándar más frecuentemente incorporado en las implementaciones comerciales de redes WPAN de bajo consumo [RHGSGSGH13], al ser adoptado total o parcialmente como especificación de los niveles físico y MAC en varios estándares que cubren otros los niveles de la pila de protocolos, como *ZigBee*, RF4CE, IP500, 6LowPAN, WirelessHART e ISA100 (en los dos últimos sólo la capa física, ya que el nivel de enlace está basado en TDMA). De estos estándares mencionados, sólo *ZigBee*, RF4CE y 6LowPAN son completamente abiertos, mientras que IP500, que está basado en el segundo, está aún en una fase poco madura de su especificación. Los estándares *ZigBee* y 802.15.4 se describirán con detalle en el capítulo 3.

El mecanismo de control de acceso al medio definido por el nivel de enlace es uno de los factores más determinantes en el consumo energético de los dispositivos en una red WPAN, resultando primordial cuando es necesario maximizar la duración de la carga de la batería. Diferentes iniciativas han propuesto alternativas a 802.15.4 para mejorar el consumo de energía mediante la reducción del ciclo de trabajo de los dispositivos, dando lugar a propuestas bien conocidas por la comunidad investigadora, e implementadas en plataformas académicas aunque no adoptadas por ningún estándar comercial, como X-MAC, S-MAC, T-MAC, B-MAC, ContikiMAC, etc. Los trabajos [BDWL10; CPMA14]

contienen una revisión de diferentes propuestas, algunas de las cuales están implementadas para las plataformas descritas más adelante en la sección 1.3.

Aunque los enlaces entre dispositivos en las redes WPAN son de corto alcance, el área de cobertura de la red puede ser ampliada si se admiten comunicaciones multisalto. El encargado de reenviar la información de un nodo a otro es el nivel de red, que además en una red radio mallada también puede ser el responsable de encontrar los posibles caminos que puede seguir la información y seleccionar el más adecuado. Diferentes mecanismos han sido sugeridos por la comunidad académica [AY05], muchos de ellos basados en propuestas bien conocidas para redes MANET como AODV y OLSR. En lo que respecta a los esfuerzos de establecer estándares abiertos para el nivel de red de las redes WPAN, el propio IEEE [LZZ+10] creó el grupo de trabajo 802.15.5 con objeto de definir un nivel de red para redes WPAN malladas basadas en 802.15.4 y adecuado para dispositivos con restricciones de memoria y reducida capacidad de proceso, aunque de momento no existen implementaciones comerciales del mismo. Otra institución que se ha involucrado en la especificación de protocolos para el nivel de red de las redes WPAN, en este caso basados en la pila de protocolos de internet IP, es el IETF. Las propuestas del IETF para las redes WPAN [ICT+13] apuestan principalmente por su integración en una red global IP versión 6, proponiendo una capa de adaptación denominada 6LowPan [SB11] para reducir la sobrecarga que conllevan las cabeceras de los paquetes IPv6. Otros grupos de trabajo del IETF han especificado también un protocolo para la búsqueda de rutas especializado en redes de bajo consumo con pérdidas denominado RPL [WTB+12].

La introducción de la capa de adaptación del protocolo IPv6 para redes de área local ha abierto nuevas posibilidades y ha favorecido que la interconexión de miles de millones de dispositivos heterogéneos a Internet se empiece a convertir en una realidad cada vez más próxima, materializando el concepto que se ha denominado Internet de las cosas (*Internet of Things*, IoT) [MSDC12]. El desarrollo de la IoT, cuya visión es que la tecnología IP sería el tejido que interconecta cualquier tipo de objeto inteligente (dispositivos vestibles, sensores, actuadores, terminales, plataformas empotradas) con otro, perseguiría facilitar el desarrollo de nuevas aplicaciones y servicios basados en el intercambio automático de información entre todos y cualquiera de los dispositivos conectados [ZBC+14; AFGM+15]. Las expectativas creadas por el inminente despliegue de la IoT son considerables y han conseguido atraer el interés de gigantes de la industria del software como IBM, Microsoft y Google, y de la industria de la microelectrónica como ARM e Intel, que han empezado a apostar por este mercado y a ofrecer plataformas y herramientas de desarrollo [WSJ15].

Otra de las líneas de trabajo abiertas por el IETF se centra en el diseño de protocolos para la capa de aplicación de dispositivos limitados en recursos, con objeto de estandarizar la forma en la que se implementan los servicios ofrecidos por los dispositivos. En este caso, la propuesta presentada por el IETF, denominada CoAP [BCS+12; SHB14], va encaminada a adaptar el exitoso modelo RESTful utilizado por los servicios web tradicionales a las limitaciones de los dispositivos y de las comunicaciones que pueden darse en un entorno WPAN, enfoque que también ha sido denominado *Web of Things* [HHP15].

No obstante, el concepto más amplio de IoT [WSJ15] no va necesariamente ligado a las redes WPAN con conectividad IPv6 directa (*seamless*), sino que también se extiende

a los procedimientos que permiten conectar entre sí redes WPAN de diversas tecnologías mediante Intenet a través de dispositivos *gateways*, permitiendo intercambiar datos entre ellas y con otras redes externas, y haciendo posible la interacción entre objetos inteligentes con conectividad WPAN (sensores, *wereables*, actuadores) y dispositivos terminales (*smartphones*, ordenadores personales) y/o servidores (bases de datos, almacenamiento en la nube, etc.) con conectividad IP. Los protocolos y tecnologías que facilitan esta integración de dispositivos y aplicaciones híbridos y heterogéneos reciben frecuentemente el nombre de *middleware*, y en muchos casos se basan en el intercambio de mensajes entre aplicaciones siguiendo un patrón de publicación/suscripción, existiendo diversas alternativas en la actualidad diseñadas con este fin como MQTT, AMQP y ZeroMQ [AFGM+15], que cuentan con una especificación abierta y diversas implementaciones tanto comerciales como de código abierto. Recientemente, los organismos reguladores internacionales (ETSI, ARIB, TIA, etc.) han mostrado también su interés en este área con objeto de permitir a los operadores de telecomunicación ofrecer acceso a estos servicios, denominados también habitualmente como M2M (Machine to Machine), de una forma armonizada [One].

En lo que respecta a las tecnologías para redes de área personal, con objeto de asegurar la completa interoperabilidad de los dispositivos algunos estándares especifican toda la pila de protocolos desde la capa física hasta el nivel de aplicación, proporcionando incluso casos de uso y pautas sobre cómo implementar determinadas aplicaciones (perfiles). Tal es el caso de las tecnologías como *ZigBee*, *Bluetooth* y *Bluetooth Low Energy* en las que se centra la presente Tesis. Estas tres tecnologías son estándares abiertos que se describen con detalle en los capítulos 2, 3 y 4. Otras soluciones estandarizadas que especifican todos los diferentes niveles de la pila de protocolos y que actualmente cuentan con una notable difusión son Z-Wave y ANT+. Ambas son soluciones propietarias y sus especificaciones no están disponibles excepto bajo suscripción a sus respectivos consorcios, aunque en el caso de Z-Wave las capas física y MAC han sido recientemente recogidas en la recomendación G.9959 del ITU-T [Sil15]. Z-Wave es una tecnología principalmente orientada a la domótica y automatización del hogar y opera en la banda libre de 868 Mhz. ANT+ por otra parte es una tecnología similar a *Bluetooth low energy* centrada en permitir la conectividad punto a punto o en estrella de dispositivos vestibles deportivos a ordenadores personales o terminales móviles.

1.2. Metodología de evaluación

La evaluación de las tecnologías inalámbricas para redes de área personal que se presenta en esta tesis, se ha llevado a cabo principalmente mediante la realización de experimentos en escenarios de prueba con dispositivos reales basados en implementaciones comerciales. Este enfoque presenta algunas ventajas e inconvenientes frente a otras técnicas habitualmente utilizadas también en estudios de investigación similares, y que se comentarán brevemente en esta sección.

Las técnicas experimentales son necesarias para analizar las prestaciones de las redes de comunicación, así como evaluar el funcionamiento de nuevas propuestas y optimizaciones, y en muchos casos también son imprescindibles para validar modelos matemáticos

del rendimiento que se hayan derivado de un análisis de su funcionamiento (ya que en muchos de estos análisis se realizan habitualmente un gran número de simplificaciones y suposiciones para poder llegar a obtener una solución computable).

Una de las estrategias habituales para obtener resultados experimentales es la realización de estudios mediante simulación. La técnica más ampliamente utilizada para ello es la simulación basada en eventos discretos, en la que el comportamiento del sistema se modela como una secuencia discreta de eventos, cada uno de los cuales tiene lugar en un momento puntual y supone un cambio de estado en el sistema, sin que se pueda producir un cambio de estado entre dos eventos consecutivos. Esto permite al simulador ir avanzando de un evento al siguiente más cercano y analizar los cambios de estado que tienen lugar en cada uno, sin tener que analizar los instantes temporales comprendidos entre uno y otro. Este tipo de simulación requiere la implementación de modelos más o menos detallados del comportamiento de los diferentes componentes del sistema basados en máquinas de estado finitas. El modelado del comportamiento de los protocolos que intervienen en la comunicación en los diferentes nodos de una red PAN y su posterior evaluación mediante una herramienta de simulación de red de propósito general como NS-2, NS-3, OMNET++ u OPNET es una de las estrategias más comúnmente utilizadas, existiendo un gran número de modelos de simulación de diferentes protocolos ya implementados para dichas herramientas. En otros casos se utilizan plataformas de simulación más específicas que se basan en que muchos de los sistemas operativos utilizados por los dispositivos que intervienen en una red PAN -y en particular en redes de sensores- siguen un paradigma de programación orientada a eventos para ejecutar en el simulador el código real de la implementación del protocolo en lugar de un modelo simplificado del comportamiento del mismo. Este es el caso de las plataformas de simulación TOSSIM [LLWC03] y COOJA [ÖDE+06], que permiten simular el comportamiento del código nativo desarrollado para las plataformas TinyOS y Contiki (que se describen más adelante) respectivamente.

La evaluación mediante experimentos realizados sobre el montaje físico de un sistema de pruebas es otra de las alternativas. La utilización de dispositivos físicos hace que el comportamiento sea en principio más realista que en los modelos de simulación, y que se incluyan efectos que al simular se modelan de forma simplificada como puede ser el caso de la transmisión por el canal radio, o efectos que habitualmente no se consideran en las simulaciones como el tiempo que puede requerir en una CPU con prestaciones limitadas el procesado de los mensajes y primitivas al propagarse a lo largo de los diferentes niveles de la pila de protocolo, e incluso otras limitaciones como las restricciones de memoria que pueden alterar el comportamiento de los nodos o que deben ser tenidas en cuenta en el diseño e implementación de los protocolos. Por otra parte esta metodología adolece de una serie de inconvenientes con respecto a las simulaciones, entre los que destaca principalmente la complejidad y el coste que puede suponer el montaje de un sistema de pruebas adecuado, lo que lleva frecuentemente a simplificar el escenario en el que se realiza la evaluación, perdiendo así parte del realismo ganado frente a las simulaciones. Otro inconveniente importante de esta metodología frente a la simulación es que es más complicado mantener todas las variables del experimento bajo control y evitar la interferencia en los resultados de fenómenos físicos no predecibles y ajenos al sistema bajo estudio. La interpretación

de los resultados obtenidos y la depuración de potenciales problemas es así mismo más compleja que al utilizar un entorno completamente controlado, en el que se puede pausar la prueba y acceder a toda la información de la red en un momento dado, como es el caso de un simulador. La coordinación de la operación de los nodos, y el establecimiento de una referencia temporal común a todos ellos (para medir parámetros como el retardo de transmisión en un sentido u OTT) puede llegar a ser un desafío importante en una red real, mientras que no lo es en absoluto en un entorno de simulación en el que el instante de simulación actual es una variable accesible por cualquiera de los modelos en todo momento. Por otra parte, la sincronización perfecta de la temporización de los nodos que se puede producir en las simulaciones hasta niveles de simultaneidad no alcanzables en la práctica también puede suponer a veces un problema que introduce diferentes distorsiones que pueden falsear los resultados (*artifacts* es el término anglosajón utilizado en la literatura para éstos fenómenos).

Las evaluación por simulación frente a la realización de experimentos reales también tiene la ventaja evidente de requerir normalmente menos tiempo para llevar a cabo el conjunto de pruebas a realizar, puesto que las operaciones no se realizan en tiempo real y las diferentes instancias de simulación correspondientes a diferentes experimentos se pueden paralelizar fácilmente mediante el uso de un *cluster* o un *grid* de computación.

Aún así, la realización de experimentos con implementaciones comerciales de los dispositivos resulta interesante en muchos casos porque permite comprobar hasta qué punto se aproxima el comportamiento real al previsto en casos más ideales, y conocer cuáles son las principales limitaciones que presenta la tecnología actual.

1.3. Plataformas para implementaciones reales

En la actualidad existen multitud de implementaciones comerciales de los diferentes estándares de arquitectura de protocolos para redes de área personal, así como plataformas de código abierto desarrolladas por la comunidad investigadora para dar soporte a la implementación de protocolos propuestos en los distintos niveles. En el caso de tecnologías completamente estandarizadas como *ZigBee* o *Bluetooth Low Energy*, diversos fabricantes de dispositivos electrónicos (transceptores y microcontroladores) como Freescale/NXP, Texas Instruments, Atmel, Microchip o Silicon Labs, ofrecen sistemas y plataformas de desarrollo propietarias para permitir la evaluación y el diseño de aplicaciones con sus productos. En la presente tesis se han evaluado principalmente estas tecnologías estandarizadas utilizando implementaciones comerciales de Texas Instruments, pero en este apartado se recogen otras alternativas que son también apropiadas para este tipo de evaluación y que cubren diferentes implementaciones de protocolos para redes WPAN.

TinyOS [LMP+05], un sistema operativo diseñado con soporte de comunicaciones para dispositivos sensores de bajo consumo, es una de las plataformas más conocidas y utilizadas habitualmente por la comunidad investigadora para la implementación de *testbeds* para redes PAN. Soporta varias plataformas *hardware* bien conocidas, como los dispositivos TelosB, MicaZ e IRIS, basadas en su mayoría en los microcontoladores MSP430 (Texas Instruments) y AtMega (Atmel), conectados a diferentes transceptores radio que operan

en las bandas libres de 868 Mhz y 2,4 Ghz. Las aplicaciones y los protocolos se implementan mediante un lenguaje especial denominado NesC [GLB+03] orientado a facilitar un desarrollo modular y una programación a alto nivel del *firmware* de los dispositivos. A pesar de su popularidad entre la comunidad académica, su enfoque está algo alejado del habitualmente utilizado en las implementaciones comerciales.

Contiki [DGV04] es otra plataforma que ofrece un sistema operativo de soporte y un conjunto de bibliotecas con implementaciones de diferentes protocolos propuestos por la comunidad investigadora o recogidos por estándares abiertos. Su diseño está principalmente orientado a habilitar aplicaciones IoT sobre dispositivos limitados en memoria y capacidad de proceso. La programación de los protocolos y aplicaciones se realiza mayoritariamente sobre C y se centra principalmente en protocolos compatibles con la pila TCP/IP (versión 4 y 6), así como en los correspondientes estándares para la adaptación de estos a redes de área personal (como 6LowPAN). Algunos de los módulos desarrollados para Contiki, como las pilas de protoclo lwIP y uIP [Dun03], son portables a otros sistemas operativos y han sido frecuentemente integrados en el *firmware* de muchos sistemas empotrados utilizados en productos y aplicaciones comerciales. El ecosistema de sistemas de desarrollo soportado por Contiki está en continuo crecimiento, soportando en la actualidad múltiples plataformas *hardware* basadas en microcontroladores y transceptores radio de diversos fabricantes.

RIOT-OS [BHG13] es un sistema más reciente de diseño similar a Contiki, y que se centra también en el paradigma de la Internet de las cosas. Actualmente se encuentra en fase de desarrollo y su madurez es menor que la de las plataformas comentadas anteriormente, siendo también menor su comunidad de usuarios.

Ante la vigente eclosión de la Internet de las cosas y las expectativas creadas, importantes compañías comerciales están empezando a implementar y ofrecer a la comunidad de desarrolladores sus propias plataformas mayoritariamente abiertas y similares en diseño a las anteriormente citadas. En octubre de 2015 ARM, compañía centrada en el diseño de arquitecturas de CPU para microcontroladores y microprocesadores, ha liberado la versión inicial su plataforma mbedOS [Arm], mientras que Intel, tras retomar con la serie de microcontroladores Quark SE una línea de productos que abandonó hace décadas, ha anunciado para finales de 2015 la publicación de una plataforma similar para el desarrollo de dispositivos vestibles basados en ellos [Int].

1.4. Objetivos y estructura de la tesis

La presente tesis se centra principalmente en el estudio de tres tecnologías de redes de área personal que pueden utilizarse para dotar de conectividad inalámbrica a objetos inteligentes: *Bluetooth*, *ZigBee/802.15.4* y *Bluetooth Low energy*. Estas tres tecnologías se han seleccionado por cumplir principalmente dos requisitos: contar con un amplio despliegue e implantación en dispositivos y sistemas comerciales, y ser estándares abiertos cuya especificación se puede consultar y para los que hay diversas herramientas de desarrollo disponibles. El estudio se ha enfocado desde la perspectiva de las posibles aplicaciones para objetos inteligentes con limitaciones de consumo y capacidad de proceso, ya que en

el paradigma de la Internet de las cosas que se plantea en la actualidad una parte importante de los objetos que pueden constituirla corresponden a sistemas de bajo coste con estas restricciones.

Todo este marco de trabajo ha conllevado que el objetivo principal de la tesis se sitúe en el estudio y evaluación de las prestaciones en términos de consumo de las tecnologías inalámbricas anteriormente mencionadas. No se trata de un estudio comparativo de las tecnologías, ya que actualmente cada una tiene un ámbito de aplicación, y en la mayoría de ocasiones no se constituyen como alternativas. No obstante, esto es algo que puede estar sujeto a cambios en un futuro, pues la evolución prevista para cada una de ellas está orientada a expandir sus nichos de aplicación.

Considerando este marco, se establecen los siguientes objetivos principales para cada una de las tecnologías bajo estudio:

- Caracterización experimental del comportamiento energético de diversas implementaciones de dispositivos comerciales.
- Desarrollo de un modelo matemático del comportamiento energético.

Si bien éstos son los objetivos principales de la tesis, en el proceso de desarrollo de los mismos para cada una de las tecnologías han surgido a una serie de objetivos secundarios, que matizan aspectos particulares o problemáticas de las tecnologías tratadas. Así, dentro de esta tesis y de forma colateral se han trabajado los siguientes objetivos asociados a cada tecnología:

- En el caso de la tecnología *Bluetooth*, la limitación del pequeño número de dispositivos que pueden interconectarse con la topología básica en estrella que impone la *piconet* puede suponer un hándicap en la formación de redes de áreas personal para determinadas aplicaciones. Por ello, se ha considerado conveniente evaluar de forma experimental el comportamiento y modos de funcionamiento de topologías que permitirían la formación de redes con un mayor número de dispositivos, denominadas *scatternets*. La formación de este tipo de redes es abiertamente permitida por el estándar pero su funcionamiento no está suficientemente definido, por lo que se ha considerado interesante comprobar el comportamiento de implementaciones comerciales en este escenario. Para ello, se hará necesario el desarrollo de una plataforma de medida de las prestaciones que permitan evaluar el comportamiento de estas topologías.
- En el caso de la tecnología *ZigBee*/802.15.4, en el análisis de cómo se ve afectado el consumo como el tiempo de sondeo (T_{poll}), se plantea el estudio de cómo este parámetro afecta también al retardo de los datos que se transmiten entre los dispositivos *end-device* de bajo consumo y los dispositivos *routers* de la red. Para ello, se desarrolla una herramienta que permite medir el retardo tanto ascendente como descendente entre dos elementos de la red.

Los capítulos 2, 3 y 4 presentan el desarrollo de estos objetivos y los resultados de la evaluación que plantean. En todos ellos se sigue la misma estructura: En primer lugar se presenta el funcionamiento de la tecnología realizando un resumen detallado de los

estándares implicados, y posteriormente se realiza una revisión del estado del arte de los trabajos de investigación relacionados con la tecnología, con especial énfasis en aquellos más relacionados con evaluaciones realizadas con un enfoque similar al escogido en esta tesis. Por último, se describe la plataforma de pruebas propuesta y se presentan los diferentes resultados obtenidos en los escenarios estudiados. Finalmente en el capítulo 5 se recogen las conclusiones de los estudios realizados, se resumen las aportaciones y contribuciones realizadas a congresos y revistas científicas, y se proponen líneas futuras de investigación.

Capítulo 2

Evaluación empírica y modelado de WPAN basadas en *Bluetooth*

2.1. Introducción

En este capítulo se presentan las aportaciones realizadas sobre la evaluación de redes basadas en la tecnología *Bluetooth*. Los fundamentos del funcionamiento de esta tecnología, derivados del estudio de sus especificaciones, se detallan en la sección 2.2, en la que se realiza una revisión completa del funcionamiento de todos los niveles del estándar. No obstante, los aspectos más relevantes para el estudio realizado en este capítulo se concentran principalmente en la sección 2.2.4.

Tras la descripción detallada del funcionamiento de la tecnología, en la sección 2.3 se realiza una revisión de los diferentes campos de estudio en los que se han centrado los trabajos y propuestas realizados por la comunidad investigadora en relación a distintos aspectos de la tecnología *Bluetooth*. Finalmente, los estudios realizados dentro del marco de la presente tesis en relación al consumo y al rendimiento de las *scatternets* formadas por dispositivos comerciales, son presentados en las secciones 2.4 y 2.5.

2.2. Fundamentos de la tecnología *Bluetooth*

Bluetooth es una tecnología inalámbrica para redes de área personal que se diseñó originalmente como alternativa a los interfaces "cableados" entre un ordenador personal y los diferentes tipos de periféricos que se pueden conectar a este, de forma que se pudieran sustituir los diferentes interfaces por un único interfaz inalámbrico, y que se posibilitara que un dispositivo móvil obtuviera la misma conectividad a periféricos y redes que un ordenador personal al integrar la tecnología *Bluetooth*.

Los objetivos iniciales de la tecnología *Bluetooth* eran por tanto facilitar las comunicaciones entre equipos móviles y fijos, eliminando los cables y conectores entre estos, ofrecer la posibilidad de crear de forma sencilla y expontánea pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales. Los escenarios para los que fue concebida condicionaron ampliamente el diseño de *Bluetooth*. Al estar orientado a la comunicación entre unos pocos dispositivos cercanos, el alcance requerido no era grande,

ni era necesario soportar redes con un gran número de dispositivos conectados, pero al estar pensada para equipos de electrónica de consumo alimentados a batería, sí que era necesario mantener limitado tanto el consumo de potencia como el coste de los transceptores. De la misma forma, puesto que la interconexión de terminales móviles con dispositivos de audio de manos libres era otro de los casos de uso contemplados, la tecnología se diseñó para soportar tanto conmutación de circuitos como de paquetes.

La primera versión considerada funcional de las especificaciones fue la 1.1, de 1999, que corregía numerosas erratas de la versión 1.0. En la versión 1.2 se realizó una reorganización de la norma, mejorando la estructura de los documentos y aclarando algunos aspectos. Además, manteniendo la compatibilidad hacia atrás se mejoraron los procedimientos de búsqueda y conexión entre dispositivos, se introdujo el mecanismo de salto de frecuencia adaptativo (AFH, *Adaptative Frequency Hopping*) para mejorar la coexistencia con otras tecnologías (particularmente con WiFi), se mejoró la calidad de las comunicaciones de voz , se introdujeron mecanismos de retransmisión de paquetes y control de flujo a nivel L2CAP, y se definieron nuevos perfiles para equipos multimedia. La versión 2.0+EDR de finales de 2004 incorporó la opción de utilizar nuevas modulaciones a nivel radio y nuevos formatos de paquete, permitiendo triplicar la tasa binaria. A mediados 2007 se publicó una nueva revisión, la 2.1+EDR, en la que se mejoran algunos procedimientos incluyendo conceptos como la respuesta extendida en el procedimiento de *inquiry*, la posibilidad de realizar *sniff subrating* para reducir el consumo y un procedimiento de emparejamiento seguro simplificado que permite un establecimiento más cómodo y rápido de comunicaciones seguras. Las revisiones posteriores de la norma, 3.0+HS de 2009 y la 4.0+LE de finales de 2011 tienen como gran novedad la introducción de nuevas tecnologías radio portadoras no compatibles hacia atrás, permitiendo grandes mejoras en la velocidad de transferencia (en el caso de la variante *High Speed* introducida en la especificación 3.0), o del consumo (en el caso de la variante *Low Energy* introducida en la especificación 4.). De esta forma en la actualidad la norma integra de momento tres tecnologías portadoras radio diferentes: la variante tradicional BR/EDR(*Basic Rate/Enhanced Data Rate*), la variante HS (*High Speed*) y la variante LE (*Low Energy*). El transporte radio *HS* requiere que el módulo controlador integre simultaneamente la portadora BR, sin embargo la variante LE puede coexistir con las otras o funcionar de forma totalmente independiente. Finalmente, la revisión 4.1 de 2013 introdujo algunas mejoras adicionales para optimizar el funcionamiento en determinados escenarios y facilitar la reconexión automática de los dispositivos sin participación del usuario. La última revisión es la 4.2 (liberada en diciembre de 2014), que ha añadido soporte para IPv6/6LoWPAN con idea de facilitar la conexión de dispositivos *Bluetooth low energy* a la internet de las cosas.

En esta sección se presenta la arquitectura de protocolos de *Bluetooth* que corresponde con el funcionamiento BR/EDR tradicional, compatible hacia atrás con las especificaciones 1.2 y 2.0+EDR. Los protocolos relacionados con la variante *low energy* de *Bluetooth*, se presentarán de forma separada más adelante en el capítulo dedicado a dicha tecnología.

2.2.1. Arquitectura de protocolos

La especificación *Bluetooth* se encuentra dividida en varios niveles o capas, como se muestra en la Figura 2.1. Por motivos de implementación la especificación distingue desde sus orígenes dos grupos diferenciados de niveles:

- Subsistema *Controller*: Formado por las niveles bajos de la comunicación: Radio, *Baseband* y *Link Manager*
- Subsistema *Host*: Compuesto por las capas altas de la pila de protocolo: L2CAP, SDP, RFCOMM, BNEP, etc.

Esta diferenciación se introduce porque normalmente los niveles más bajos de la pila de protocolos residen en un dispositivo *hardware* específico, denominado *Bluetooth controller*, mientras que los niveles superiores se implementan como componentes *software* que se ejecutan en un microprocesador de propósito general. El *Bluetooth controller* o módulo *Bluetooth* suele ser un *System on Chip* (SoC) que integra el subsistema radio con un controlador empotrado cuyo *firmware* implementa el resto de protocolos de nivel inferior. Este controlador empotrado se conecta a través de un interfaz *hardware* a un microprocesador de propósito general (*host*), en el cual residen los protocolos de nivel superior que se implementarían como componentes de los protocolos de red del sistema operativo del *host*. La especificación *Bluetooth* establece un interfaz normalizado denominado HCI ($Host-Controller\ Interface$) que define cómo se deben comunicar las capas software de nivel superior con el *firmware* integrado en el módulo *Bluetooth*, identificando además diferentes posibilidades de transporte o interfaces *hardware* para interconectar el dispositivo *controller* con el *host* (USB, SPI, UART).

A pesar de esta separación establecida por la norma, también pueden encontrarse en el mercado módulos radio *Bluetooth* de pila completa, que integran un subconjunto de los protocolos de ambos grupos para implementar un perfil específico.

Aunque en la versión 1.1 de las especificaciones sólo se distinguía entre los niveles Radio, *Baseband* y *Link Manager*, a partir de la versión 1.2 se realizó una subdivisión de estos niveles definiendo nuevos bloques arquitecturales, que se muestran también en la figura 2.1: *Device Manager*, *Base Band Resource Manager*, *Link Controller* y *Link Manager*. Esta subdivisión en bloques del controlador *Bluetooth* BR/EDR se ha mantenido en las diferentes revisiones de las especificaciones hasta la fecha.

No todos los protocolos adoptados por el SIG *Bluetooth* forman parte del núcleo (*Core*) de las especificaciones. Dicho núcleo está compuesto por los protocolos del controlador (Radio, *Baseband* y *Link Manager*) más los protocolos L2CAP y SDP. La especificación de otros protocolos adoptados, como BNEP, RFCOMM, AVCTP o AVDTP se recogen en documentos separados del *Bluetooth Core Specification*, y por tanto siguen una numeración de versiones diferente. En algunos perfiles de *Bluetooth* se integran también en los niveles superiores protocolos no adoptados por el SIG, que son especificados y mantenidos por otras organizaciones (como por ejemplo PPP, IP o TCP/UDP, descritos en RFCs del IETF).

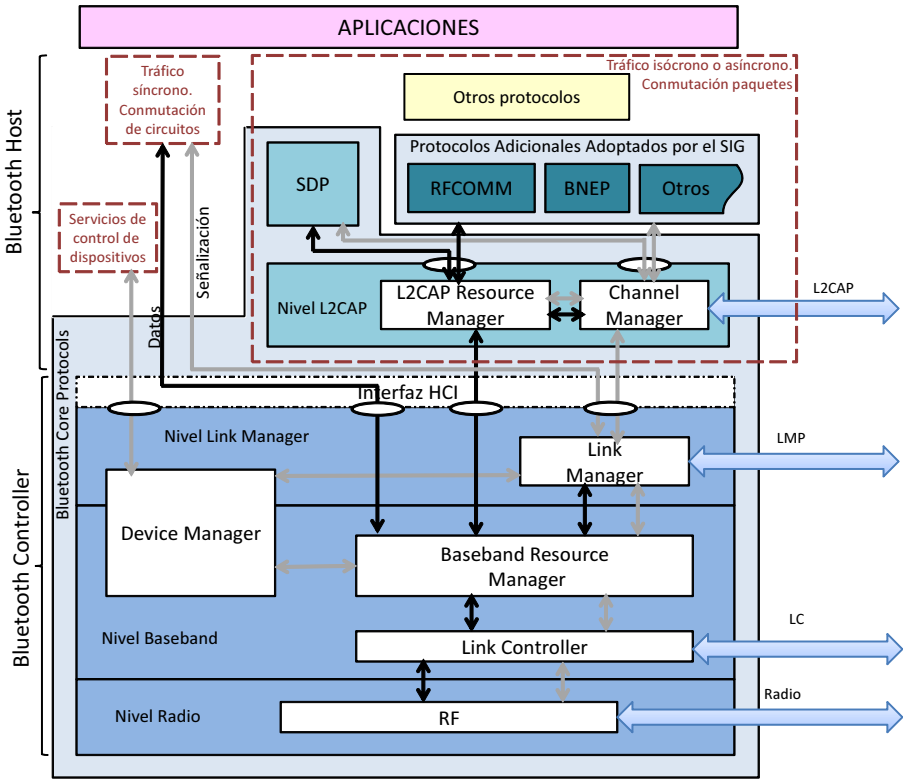


Figura 2.1: Arquitectura de Protocolos Bluetooth

2.2.2. Nivel Radio

Los dispositivos que usan tecnología *Bluetooth*, utilizan la banda libre denominada ISM (*Industrial Scientific Medical*). Al ser una banda libre, es necesario tener en cuenta que se producirán interferencias con otros dispositivos que la usen, e incluso con distintas redes *Bluetooth*, por lo que es necesario utilizar algún mecanismo que reduzca lo máximo posible esta situación. Para conseguir dicho objetivo, el estándar *Bluetooth* realiza cambios periódicos de la frecuencia de transmisión conforme a una secuencia de saltos seudoaleatoria (1600 saltos/seg), dentro de un rango de canales de 1 MHz cada uno. Esta secuencia de saltos es la misma en todos los dispositivos de una misma red *Bluetooth*, formada por un maestro y sus esclavos, y es controlada por la capa *Baseband*.

El nivel radio define cómo se codifica la información para dividir la banda disponible en canales que puedan ser utilizados por los diferentes dispositivos. Actualmente se definen dos modos de modulación, el modo básico (BR, *Basic Rate*), de implementación obligatoria, y el modo de tasa mejorada (EDR, *Enhanced Data Rate*), cuya implementación es opcional. En el primer modo, que es compatible hacia atrás con las especificaciones 1.1 y 1.2, la técnica de modulación es GFSK (*Gaussian Frecuency Shift-Keying*) con una tasa binaria de 1 Mbps. En el segundo modo, incorporado de forma opcional a partir de la versión 2.0 de las especificaciones, se utilizan dos técnicas de modulación adicionales, $\pi/4 - DQPSK$ (*Differential Quaternary Phase Shift Keying*) y $8DPSK$ (*Differential Phase Shift Keying*), con tasas binarias de 2 Mbps y 3 Mbps respectivamente. En todos los casos el ancho de banda del canal radio es de 1 MHz, dividiendo la banda ISM de 2.4

GHz en un total de 79 canales radio.

Bluetooth define varias clases de dispositivos radio en función de su potencia máxima de emisión, lo que a su vez determina el alcance que pueden conseguir:

- Radio Clase 3: Con una potencia de emisión máxima de 1mW (0 dBm).
- Radio Clase 2: Es la más común en dispositivos móviles y tiene un alcance aproximado de 10 metros, con una potencia de emisión entre 0,25 y 2,5 mW (-6 a 0 dBm), control de potencia opcional y una potencia de emisión por defecto de 1 mW (0 dBm).
- Radio Clase 1: De uso principalmente industrial, tiene un alcance aproximado de 100 metros y una potencia de emisión entre 1 y 100 mW (0 a 20 dBm), con control de potencia obligatorio.

2.2.3. Nivel Baseband

El nivel *Baseband* define cómo se utilizan los canales radio para la transmisión de información entre los dispositivos, determinando en gran medida las características fundamentales y los procedimientos básicos de la tecnología Bluetooth.

El sistema *Bluetooth* proporciona una conexión punto-punto o una conexión punto-multipunto, en donde los dispositivos en cuestión comparten el canal. Dos o más unidades compartiendo el canal forman una *piconet* (Figura 2.2). En cada *piconet* existe una unidad *Bluetooth* que actúa como "maestro" y el resto (hasta 7 dispositivos) actúan como "esclavos". Se utiliza un esquema de división en el tiempo TDD (*Time Division Duplex*) para que maestro y esclavos transmitan alternativamente sus paquetes. En una *piconet* puede haber hasta siete esclavos activos, aunque pueden existir muchos más en modo *park* (modo en el que no están activos en el canal, pero siguen sincronizados con el maestro). El acceso al canal, tanto de esclavos activos como "aparcados" está controlado por el maestro de la *piconet*.

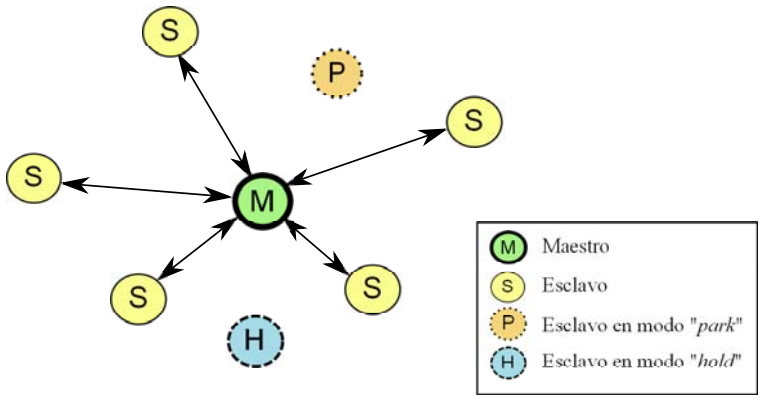


Figura 2.2: *piconet Bluetooth*

Varias *piconets* que comparten dispositivos entre sí, forman una *scatternet* (Figura 2.3). Cada *piconet* tiene un único maestro, que a su vez puede ser esclavo en otra *piconet*.

Los esclavos pueden participar en varias *piconets* para lo cual se utiliza multiplexación en el tiempo. Los dispositivos que están participando de forma simultánea en más de una *piconet*, bien como maestro en una y esclavo en otra, o bien como esclavo en dos o más, son denominados *bridges*. Cada *piconet* tiene su propio canal, definido por una secuencia de saltos en frecuencia como se detallará más adelante, que depende exclusivamente de la unidad que actúa como maestro de esa *piconet*.

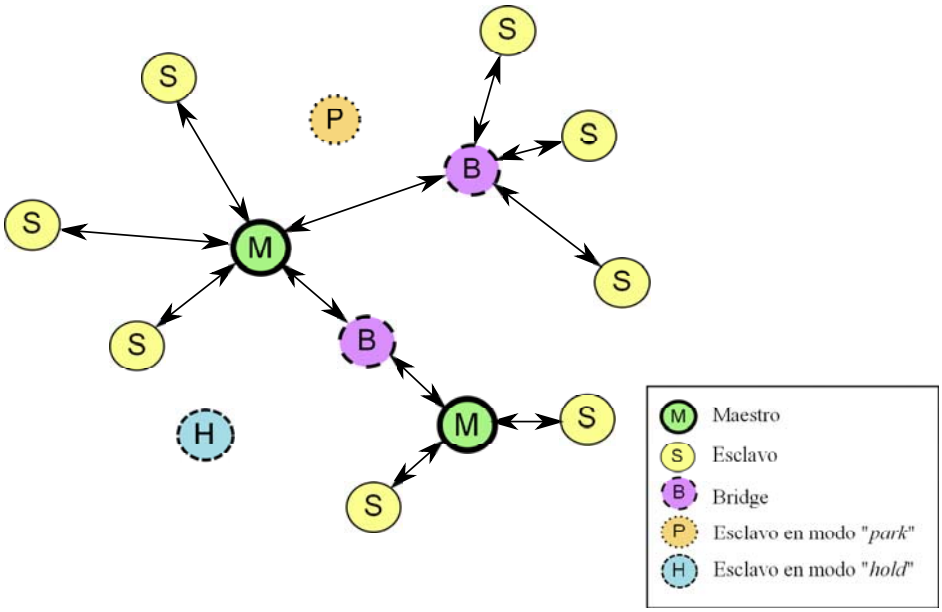


Figura 2.3: *Scatternet* Bluetooth

Como ya se mencionó al presentar la arquitectura *Bluetooth*, desde de la versión 1.2 hasta la actualidad la norma distingue tres entidades en el nivel *Baseband*:

- *Device Manager*: Es el módulo que controla el comportamiento general del dispositivo Bluetooth, y es responsable de todas aquellas funciones no directamente relacionadas con el transporte de datos, sino con la gestión de las comunicaciones, tales como la búsqueda de dispositivos y el establecimiento de conexiones. Para llevar a cabo estas funciones solicita al subsistema de gestión de recursos *Baseband Resource Controller* acceso al medio radio. También gestiona algunos comandos HCI relacionados con la configuración y parámatros del dispositivo y el almacenamiento de claves.

- *Baseband resource manager*: Es el subsistema responsable de controlar el acceso al medio radio, realizando dos funciones principales: negociar los parámetros de calidad de servicio con las diferentes entidades que van a compartir el medio físico y llevar a cabo una planificación de las comunicaciones con objeto de cumplir los criterios previamente establecidos.

- *Link Controller*: Es el subsistema responsable de implementar la codificación y descodificación de los paquetes que se envían por el canal radio, y gestiona la señalización relativa a retransmisiones y control de flujo.

2.2.3.1. Arquitectura del transporte de datos

El concepto de la arquitectura de transporte de datos se introdujo en la versión 1.2 del estándar *Bluetooth* para mejorar la consistencia y legibilidad de la norma como parte de la reestructuración llevada a cabo en dicha versión, y se ha mantenido hasta la actualidad. El sistema de transporte de datos sigue una arquitectura estructurada en capas, como se muestra en la figura 2.4, distinguiendo la capa física, la capa lógica y la capa L2CAP.

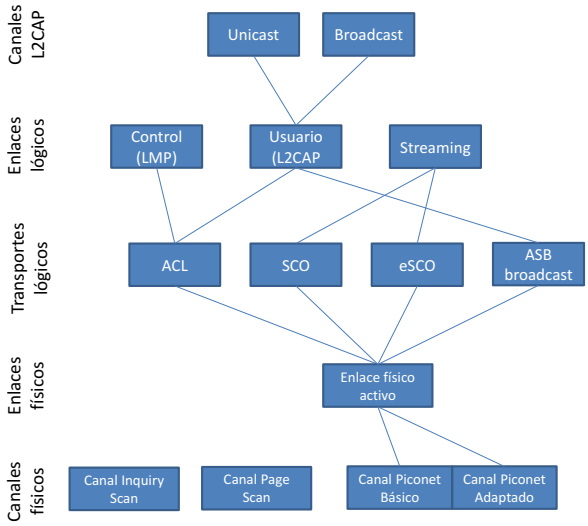


Figura 2.4: Arquitectura del transporte de datos en Bluetooth

Por cuestiones de eficiencia y compatibilidad la arquitectura de transporte establece una subdivisión de la capa lógica, distinguiendo entre el transporte lógico y el enlace lógico. El enlace lógico proporciona un canal de datos independiente entre dos dispositivos. La introducción de la capa de transporte lógico es necesaria para definir la interdependencia entre algunos de los tipos de enlaces lógicos, principalmente debida a compatibilidad hacia atrás. Anteriormente a la versión 1.2 de la especificación, la norma describía los enlaces ACL y SCO como enlaces físicos. Sin embargo, tras la adición del SCO extendido (eSCO), y de cara a futuras ampliaciones, resultó más conveniente reformular su denominación como transportes lógicos.

Puesto que la terminología utilizada por la norma puede inducir cierta confusión, es necesario aclarar que los canales físicos, enlaces físicos, transportes lógicos y enlaces lógicos son diferentes niveles definidos por la capa *Baseband*, apoyándose a su vez en los canales definidos por la capa radio (canales radio). Los enlaces lógicos definidos por la capa *Baseband* dan a su vez servicio a las capas superiores (L2CAP en el caso del transporte de datos y *Link Manager* en el caso de señalización), aunque su establecimiento, configuración y liberación es controlada por el *Link Manager*. De forma resumida y algo simplificada, los canales y enlaces físicos definen las secuencias básicas de salto de frecuencia a utilizar y la temporización del esquema de acceso múltiple, los transportes lógicos definen cómo se gestionan los recursos creados por el esquema de acceso múltiple entre las diferentes

comunicaciones, y los enlaces lógicos definen el tipo de información que se transmite entre las entidades y el plano al que pertenece.

2.2.3.2. Esquema de acceso múltiple

El esquema de acceso múltiple permite dividir en secciones accesibles el ancho de banda del canal radio, de forma que pueda ser compartido por diferentes comunicaciones. En el caso de *Bluetooth*, el acceso de diferentes dispositivos al medio radio se garantiza mediante un esquema de división en el tiempo (TDMA, *Time Division Multiple Access*) con salto de frecuencia (FH, *Frequency Hoping*).

El maestro de cada *piconet* establece una secuencia pseudoaleatoria de saltos de frecuencia, constituyendo un canal compartido entre los diferentes dispositivos conectados a la *piconet*. Este canal es compartido entre los dispositivos que constituyen la *piconet* mediante la división del tiempo en intervalos temporales o *time-slots* de 625 μs de duración, de forma que cada dispositivo puede transmitir únicamente en determinados intervalos de tiempo y en cada uno de ellos sólo transmite un único dispositivo. Para que la comunicación entre el maestro y los esclavos pueda ser bidireccional, se utiliza un esquema TDD (*Time Division Duplex*), que determina que la comunicación desde el maestro hacia los esclavos y la transmisión en sentido esclavo-maestro se realizan alternativamente en diferentes intervalos de tiempo, fijando que la comunicación maestro-esclavo comience en los *slots* pares, mientras que en sentido contrario comienza en los *slots* impares.

La transmisión en cada intervalo temporal se realiza a una frecuencia distinta, siendo la tasa de salto de frecuencia de 1600 cambios por segundo. La técnica de salto de frecuencia (FH) facilita la coexistencia de diferentes *piconets* en un mismo entorno, ya que permite compartir el espectro asignado al sistema sin tener que realizar una planificación de frecuencias. Esto es así porque la secuencia de salto de frecuencia es aleatoria y dependiente de la dirección física del maestro (BD-ADDR), que es distinto en cada *piconet* por lo que la probabilidad de que dos *piconets* transmitan a la misma frecuencia de forma simultánea es baja. Además, cuando se produce una colisión debida a que dos o más dispositivos transmiten a la misma frecuencia ésta sólo afecta a un time-slot concreto, ventaja a la que se añade la incorporación de la diversidad frecuencial inherente a las técnicas de espectro ensanchado. Por otra parte, este sistema presenta la desventaja evidente de que, al no realizar planificación de frecuencias, las prestaciones pueden degradarse apreciablemente al aumentar el número de *piconets* que se interfieren mutuamente.

Los dispositivos *Bluetooth* siguiendo el estándar IEEE 802, adoptan una dirección física única (BD-ADDR) de 48 bits, la cual está compuesta por los campos siguientes:

- LAP (*Lower Address Part*): Compuesta por los 24 bits menos significativos de la BD-ADDR.
- UAP (*Upper Address Part*): Compuesta por los 8 bits siguientes a la LAP.
- NAP (*Non-significant Address Part*): Compuesta por los siguientes 24 bits.

2.2.3.3. Canales físicos

Siguiendo el esquema general de acceso al múltiple descrito anteriormente, la norma define el concepto de canal físico. Cada canal físico viene determinados por la combinación de una secuencia de saltos pseudoaleatoria concreta, la temporización exacta del *slot* de transmisión, y el código de acceso utilizado en la cabecera del paquete (que se describirá más adelante en esta misma sección). Cuando un dispositivo se encuentra sincronizado en tiempo, frecuencia y código de acceso con un determinado canal se dice que está conectado a dicho canal.

La norma define 4 tipos de canales físicos, cada uno de los cuales está orientado a un uso específico:

- *Basic piconet physical channel*
- *Adapted piconet physical channel*
- *Page Scan physical channel*
- *Inquiry Scan physical Channel*

Los canales *basic piconet channel* y *adapted piconet channel* se utilizan para la comunicación entre dispositivos conectados y están asociados con una piconet determinada. El *basic piconet channel* utiliza una secuencia pseudoaleatoria de saltos entre un total de 79 canales, que, como ya se ha mencionado, viene definida por la dirección del dispositivo maestro, estando la fase en dicha secuencia determinada por su reloj. Todas las unidades *Bluetooth* que participan en una *piconet* están sincronizadas tanto en tiempo como en salto con el canal. El *adapted piconet channel* fue introducido a partir de la versión 1.2 de las especificaciones como parte del mecanismo AFH (*Adaptative Frequency Hopping*), que mejora la coexistencia con otras redes al permitir la eliminación de alguno de los 79 canales radio de la secuencia de salto hasta dejarlos en un mínimo de 20 canales radio. Por lo demás, su comportamiento y temporización son idénticos a los del canal básico.

Los canales *page scan physical channel* e *inquiry physical channel* se emplean en los procedimientos de búsqueda de dispositivos y de conexión que se detallarán más adelante, utilizan un subconjunto de los canales radio disponibles y una temporización diferente a la del canal básico. Esta temporización se detallará más adelante en esta sección cuando se describan los procedimientos de búsqueda de dispositivos (*Inquiry*) y conexión (*Page*).

Un dispositivo *Bluetooth* sólo puede utilizar uno de los cuatro canales en un instante dado, aunque la concurrencia de operaciones (conexión a varias *piconets*, búsqueda y conexión con otros dispositivos mientras se está participando en una *piconets*) se puede seguir soportando mediante división en el tiempo.

2.2.3.4. Enlaces físicos

La definición de los enlaces físicos fue modificada por la versión 1.2 respecto a lo especificado en la versión 1.1 del estándar, y se ha mantenido desde entonces. Un enlace físico representa una conexión de banda base entre dispositivos y está asociado con un

canal físico. Los enlaces físicos poseen características comunes que se aplican a todos los transportes lógicos que vayan sobre el enlace físico. Las características comunes de los enlaces físicos son:

- Control de potencia.
- Supervisión del enlace.
- Cifrado y Encriptación.
- Adaptación de la tasa de datos a las condiciones del canal.
- Control de paquetes *multi-slot*.

La supervisión del enlace es fundamental puesto que una conexión puede perderse debido a diversas razones, como pérdida de rango, interferencias severas o fallo de potencia. Puesto que esto ocurre generalmente sin previo aviso, es importante monitorizar el enlace en ambos extremos (maestro y esclavo). Para detectar la pérdida del enlace, tanto el esclavo como el maestro utilizan un temporizador de supervisión $T_{Supervision}$. Cada vez que se recibe un paquete válido destinado a uno de los esclavos del enlace físico el temporizador se reinicia, y si el temporizador alcanza su valor máximo se considera que se han desconectado. Este mismo temporizador se utiliza para los transportes lógicos SCO, eSCO y ACL. El valor de este tiempo máximo, *supervisionTO* es negociado por el *Link Manager* en función de los parámetros y modos de bajo consumo establecidos en la conexión, para evitar que la desconexión se dispare accidentalmente.

2.2.3.5. Transportes lógicos

Existen cinco tipos de transportes lógicos, que pueden establecerse entre maestro y esclavo.

- Transporte Lógico SCO *(Synchronous Connection-Oriented)*.
- Transporte Lógico ACL (*Asynchronous Connection-Less*).
- Transporte Lógico eSCO (*Extended Synchronous Connection-Oriented*).
- Transporte Lógico ASB (*Active Slave Broadcast*).
- Transporte Lógico PSB (*Parked Slave Broadcast*).

Los transportes lógicos SCO son simétricos y de tipo punto-a-punto entre el maestro y un esclavo de la *piconet*, para los que el maestro reserva *slots* de forma periódica, ofreciendo una comunicación por conmutación de circuitos orientada a soportar tráfico con temporización crítica (típicamente voz). El maestro puede soportar hasta tres enlaces SCO con el mismo o distintos esclavos. Cada esclavo puede a su vez soportar hasta tres enlaces SCO con el mismo maestro, o dos enlaces SCO si éstos proceden de maestros distintos. Debido al tipo de tráfico para el que están pensados, los paquetes SCO no utilizan retransmisión.

El transporte lógico eSCO puede ser simétrico o asimétrico y es también de tipo punto-a-punto entre el maestro y un esclavo. Son similares a los transportes lógicos SCO y pueden considerarse, del mismo modo, del tipo conmutación de circuitos. Además de los *slots* dedicados, el transporte lógico eSCO puede tener una ventana de retransmisión después de dichos *slots*.

En los *slots* que no están reservados para transportes lógicos síncronos, el maestro puede intercambiar paquetes con cualquier esclavo de la *piconet*. El transporte lógico ACL es también un enlace punto-a-punto entre el maestro y los esclavos de la *piconet*, y puede ser establecido junto con un enlace SCO, con un determinado dispositivo. Solo puede existir un enlace ACL entre un maestro y un esclavo. Un esclavo puede transmitir un paquete ACL en un *slot* esclavo-maestro (*slot* impar) solo si ha sido direccionado por el maestro en el *slot* maestro-esclavo (*slot* par) anterior. Este tipo de comunicación es considerada conmutación de paquetes y soporta servicios tanto asíncronos como isócronos. Para la mayoría de los paquetes ACL se utiliza retransmisión para garantizar la integridad de los datos.

El maestro de la *piconet* utiliza el transporte lógico ASB para comunicarse unilateralmente con todos los esclavos activos (es decir, se trata de una comunicación asimétrica punto a multipunto). Es un transporte no confiable, y para mejorar ligeramente la fiabilidad se retransmite cada paquete un determinado número de veces. El número de secuencia será el mismo en todas las retransmisiones para facilitar el filtrado en el esclavo destino. El maestro puede enviar estos paquetes en cualquier momento.

El maestro de la *piconet* usa el transporte lógico PSB para la comunicación con esclavos en estado aparcado. Este transporte lógico es algo más complejo que el resto, ya que consta de una serie de fases, cada una de ellas con un objetivo diferente, que se detallarán más adelante.

A cada esclavo en modo activo en una *piconet* se le asigna una dirección transporte lógico LT-ADDR primaria de 3 bits, que es incluida en la cabecera de los paquetes que se envían para identificar a qué esclavo corresponden. El maestro no requiere dirección LT-ADDR, reservándose el valor 0 para los mensajes *broadcast* del transporte ASB. La definición de LT-ADDR sustituyó al concepto de dirección AM-ADDR (*Active Member Address*) definido originalmente por la versión 1.1 de la norma. Adicionalmente al esclavo se le asigna una dirección LT-ADDR secundaria por cada transporte lógico eSCO establecido, que sólo podrá ser utilizada para enviar paquetes pertenecientes a dicho tipo de tráfico. Las direcciones LT-ADDR asignadas a un esclavo tienen validez mientras este no se desconecte o entre en modo *park*. Cuando un dispositivo se une a una *piconet* pasa a ser identificado por su dirección MAC (BD-ADDR) o por una dirección de 8 bits asignada por el maestro denominda PM-ADDR (*Park Member Address*). Además el maestro también le asigna en este caso una dirección de petición de acceso (AR-ADDR) que no es única y se utiliza para determinar la temporización de su comunicación con el maestro, como se detalla en la sección 2.2.4.4.

2.2.3.6. Formato de los paquetes *Baseband*

El paquete que un dispositivo intercambia con otro durante un intervalo temporal determinado es una ráfaga de bits constituida por varios campos. La estructura general se muestra en la figura 2.5 y es válida tanto para el modo básico BR como para el modo extendido EDR. En cualquier paquete se pueden distinguir los siguientes campos:

- El **Código de acceso al canal**, que se utiliza para facilitar la sincronización de la radio de los diferentes dispositivos y para identificar y/o filtrar las comunicaciones asociadas a un canal físico determinado. Todos los paquetes enviados por el mismo canal físico son precedidos por el mismo código de acceso. Existen cuatro tipos de códigos de acceso: El *Channel Access Code* (CAC) identifica a la *piconet* y encabeza a todos los paquetes intercambiados por los dispositivos conectados a la *piconet*, el *Device Access Code* (DAC) se utiliza en procedimientos de señalización especiales, y finalemente el *General Inquiry Access Code* (GIAC) y los *Dedicated Inquiry Access Codes* (DIACs) se utilizan en los procedimientos *Inquiry* general y dedicado, respectivamente. El CAC tiene una longitud de 72 bits, mientras que el DAC, GIAC, y DIAC pueden tener una longitud entre 68 y 72 bits dependiendo de si el paquete cuenta o no con más campos.

- La **Cabecera** *Baseband* contiene información de control del enlace y consta de 6 campos que se muestran en la figura 2.5:

  - LT-ADDR (3 bits): Es la dirección del transporte lógico correspondiente al paquete. Hay que destacar que por compatibilidad con la versión 1.1, en la que este campo se denominaba AM-ADDR, la dirección del transporte lógico se reutiliza en los transportes SCO y ACL del mismo esclavo, por lo que no es suficente con este valor pra identificar el transporte lógico, siendo necesario para ello utilizar conjuntamente este campo y el campo tipo de paquete.
  - $TYPE$(4 bits): Identifica el tipo del paquete, permitiendo distinguir entre 16 subtipos. El significado de este campo depende del transporte lógico, por lo que tiene que ser interpretado junto con el identificador del transporte lógico para determinar el tipo de paquete y de transporte. El tipo de paquete determina cuantos *slots* ocupa el paquete y la codificación empleada, determinando a su vez el tamaño máximo del *payload* que puede transportar.
  - FLOW: Bit de Control de flujo para paquetes ACL.
  - ARQN: Bit que se utiliza para reconocer/rechazar los paquetes transmitidos en sentido contrario, lo que permite recuperar los errores mediante retransmisiones automáticas. El protocolo de recuperación de errores que se utiliza es del tipo *Stop-and-Wait*.
  - SEQN: Número de secuencia que permite reordenar los datos. Sólo es de un bit, ya que el protocolo ARQ que se utiliza es del tipo *Stop and Wait*.
  - HEC: Información de control para detección de errores en la cabecera.

- El **payload** o campo de datos es el campo en el que se encapsulan los paquetes de nivel superior, que pueden ser el LMP (Link Manager Protocol) o el L2CAP (Logical Link Control and Adaptation Protocol). El *payload* es un campo que sólo existe en algunos tipos de paquete, y según dicho tipo puede transmitirse utilizando distintos niveles de codificación FEC y/o modulación.

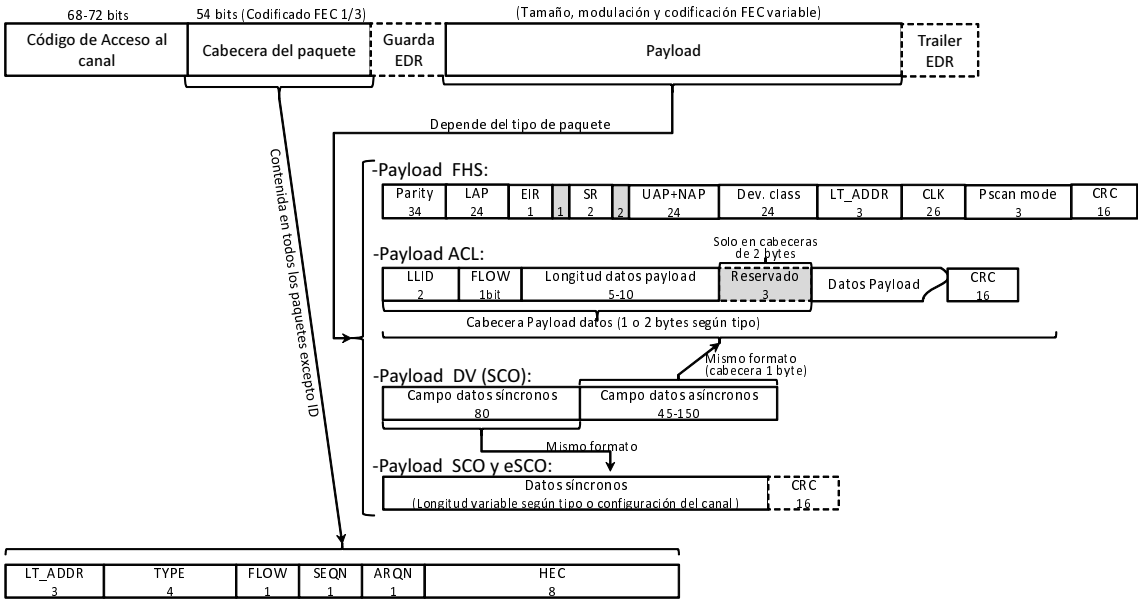


Figura 2.5: Formato de los paquetes *Baseband*. El tamaño de los campos se indica en bits.

En los paquetes ACL el *payload* consta a su vez de varios campos:

- Cabecera del payload, que consta de:
  - Un campo LLID (*Logical Link Identificator*) de 2 bits, que permite distinguir la información del plano de control ACL de la información del plano de usuario, y que sólo está presente en los transportes lógicos que soportan enlaces lógicos ACL. Además, en el caso de datos del plano de usuario, que pueden enviarse en uno o varios paquetes, este campo permite identificar si se trata del primer segmento (que por tanto debe contener la cabecera de nivel superior L2CAP) o de una continuación.
  - Un campo de control de flujo de 1 bit,
  - Un campo de longitud de 5 o 10 bits que indica la longitud del campo de datos del payload. La longitud de este campo depende del tipo de paquete, haciendo que la cabecera del payload tenga una longitud de 1 o 2 bytes según el caso.
  - Un campo reservado de 3 bits que se introduce únicamente cuando la longitud de la cabecera del payload es de 2 bytes.
- Un campo de datos de nivel superior (L2CAP o LM) de longitud variable.
- CRC de 2 bytes para verificar la integridad de los datos del *payload*.

En el caso de los paquetes transmitidos utilizando las modulaciones EDR, se establece un tiempo de guarda y sincronización adicional entre la cabecera y el payload, y un trailer tras éste, que no existen en el caso de los paquetes BR. Esto se hace para mantener la compatibilidad hacia atrás con dispositivos que no soportan EDR, permitiendo que estos participen en una *piconet* coexistiendo con otros dispositivos que sí soportan este modo. Esto se consigue forzando que la cabecera de todos los paquetes, incluidos los paquetes EDR, se codifique utilizando la modulación básica GFSK, utilizando la modulaciones $4/\pi - DPSK$ y $8DPSK$ únicamente para codificar el *payload* de los paquetes EDR. El tiempo de guarda entre la cabecera y el *payload* en estos paquetes se introduce para dar tiempo a conmutar de modulación y sincronizar el demodulador.

2.2.3.7. Tipos de paquetes

Los tipos de paquetes definidos por el nivel *Baseband* dependen del transporte lógico y de los enlaces físicos en los que pueden ser utilizados. Para distinguir diferentes tipos de paquetes en un mismo transporte lógico se utiliza el campo de 4 bits *Type* de la cabecera, lo que permitiría definir hasta 16 tipos de paquete por transporte lógico, si bien actualmente no todos los valores son válidos.

Los diferentes tipos de paquetes se definen según la naturaleza de la información que transportan, la codificación y protección contra errores empleada, y la duración de los mismos, que debe ser un número impar de time-slots (1,3 o 5), ya que según la temporización impuesta por el esquema TDD, la transmisión en sentido maestro-esclavo debe comenzar siempre en *slots* pares mientras que en sentido contrario debe comenzar en *slots* impares.

Los tipos de paquetes están divididos en varios segmentos o grupos: El primer segmento está formado por paquetes de control, que ocupan un sólo *time slot*. El segundo segmento lo forman paquetes que ocupan un *time slot*, el tercero paquetes de tres time-slot y el cuarto paquetes de cinco time-slots. En los segmentos 2, 3 y 4, los paquetes definidos son específicos para cada transporte lógico (SCO, eSCO, ACL-U y ACL-C), mientras que los paquetes del primer segmento son comunes. Los paquetes ACL están sometidos al control de flujo y pueden ser retransmitidos automáticamente si no son reconocidos por el destino. Por el contrario, los paquete SCO no llevan CRC para detección de errores y no son retransmitidos, aunque sí utilizan codificación FEC para aumentar la robustez de la comunicación. Los Paquetes eSCO sí llevan CRC.

A continuación se incluye un resumen de los principales tipos de paquete definidos por la norma:

- **Paquetes comunes de control**, de los que hay 5 tipos:

  - **Paquete ID** o paquete de identificación. Tiene una longitud fija de 68 bits, y no lleva cabecera (por lo que no dispone de campo *TYPE*) ni payload, sólo el código de acceso (DAC - *Device Access Code*) *Bluetooth* o en el código de acceso de búsqueda (IAC - *Inquiry Access Code*). Es utilizado por los procedimientos de *Paging*, *Inquiry* y *Response*.

- **Paquete NULL**. Está compuesto únicamente por el código de acceso al canal (*Access Code*) y la cabecera, no tiene campo de datos. No necesita ser confirmado por el otro extremo. Se utiliza para devolver a la unidad origen información de enlace o el estado del *buffer* de recepción. Tiene una longitud fija de 126 bits.
- **Paquete POLL**. Es muy similar al paquete NULL, ya que tampoco tiene campo de datos, aunque en este caso sí requiere confirmación por parte del otro extremo. El maestro de la *piconet* utiliza este tipo de paquete para realizar un proceso de *polling* a los distintos esclavos que participan en la *piconet*, los cuales están obligados a responder incluso si no tienen información que transmitir, a menos que el *slot* que se emplee para ello se encuentre ocupado. Dicha situación sólo puede darse si existe una red mayor que la *piconet* construida, en este caso, denominada *scatternet*.
- **Paquete FHS (*Frequency-Hopping Selection*)**. Se trata de un paquete de control que contiene entre otras cosas la dirección y la temporización (reloj) del dispositivo *Bluetooth* transmisor. El paquete FHS ocupa un solo *slot*. Es un paquete que necesita ser confirmado por el otro extremo. Se utiliza en los procedimientos de *Page*, *Inquiry* y *Role-Switching*. El paquete FHS contiene información en tiempo real del reloj del transmisor la cual permite la sincronización de la secuencia de saltos en frecuencia antes de que se haya establecido el canal de la *piconet*, o bien cuando una *piconet* existente cambie a una nueva *piconet*.
- **Paquete DM1**. Permite el envío de mensajes de control en cualquier tipo de enlace. Además de esto puede contener datos de usuario codificados con codificación FEC de 2/3, contando con un payload de 17 bytes.

- **Paquetes asociados al transporte lógico ACL.**

  - **Paquetes BR**. Son aquellos paquetes que ya estaban definidos en la versión 1.2 del estándar y que utilizan sólo la modulación GFSK. Existen diversos tipos según el número de *slots* (1,3 o 5) que ocupan y del tipo de codificación FEC empleado. Los paquetes DH1, DH3, y DH5 no utilizan codificación FEC, ocupan 1, 3 y 5 *slots* y tienen un payload de 341, 185 y 27 bytes respectivamente. Los paquetes DM3 y DM5 utilizan codificación FEC 2/3 y tienen un payload de 123 y 226 octectos respectivamente.
  - **Paquetes EDR**. Son aquellos incorporados en la versión 2.0 y posteriores, cuyo campo payload va codificado con las modulaciones EDR y sin FEC. Los paquetes 2-DH1, 2-DH3 y 2-DH5 utilizan la modulacións $4/\pi QPSK$, teniendo un payload de 56, 369 y 681 bytes, mientras que los paquetes 3-DH1, 3-DH3 y 3-DH5 utilizan la modulación 8DPSK y tienen *payloads* de 85, 554 y 1023 bytes respectivamente.

- **Paquetes asociados al transporte lógico SCO**: Se definen los paquetes HV1, HV2, HV3 y DV. En este caso el número de terminación no indica los *slots* que

ocupan, ya que todos ocupan 1 *slot*, sino la intensidad de la codificación FEC (1/3 para el HV1, 2/3 para el HV3 y ninguna para el HV3, dando un payload de 10, 20 y 30 bytes respectivamente). Los paquetes DV se utilizan para transmitir 80 bits de datos síncronos y hasta 150 bits de datos asíncronos, estando éste último campo codificado con FEC 2/3.

- **Paquetes asociados al transporte lógico eSCO**: Según el tipo de modulación, la duración en *slots* y la codificación FEC empleada se distinguen diferentes tipos de paquetes. Los paquetes EV3, EV4 y ·EV5 utilizan la modulación básica, ocupando el primero 1 *slot* y los dos últimos 3 *slots*, y utilizando el EV4 codificación FEC 2/3. Los paquetes 2-EV3, 2-EV5, 3-EV3 y 3-EV5 son idénticos a los anteriores, pero utilizando las modulaciones extendidas.

2.2.3.8. Enlaces lógicos

En *Bluetooth* BR/EDR se definen cinco tipos de enlace lógico, de los que tres son de usuario y dos son de control:

- Enlaces del plano de control

  - LC (*Link Control Logical Link*): Se incluye en la cabecera de control de todos los paquetes excepto el paquete ID. Transporta tanto información de control lógico, como de control de flujo y retransmisión, y caracterización de los datos.
  - ACL-C (*ACL Control Logical Link*): Se utiliza para el transporte de información entre las entidades *Link Manager* del maestro y del esclavo. Para ellos se utilizan paquetes de tipo DM1 o DV y se soporta tanto sobre el transporte lógico SCO como sobre el transporte lógico ACL.

- Enlaces del plano de usuario

  - ACL-U (*User Asynchronous/Isochronous*). Se utiliza para enviar paquetes de datos asíncronos o isócronos correspondientes al nivel superior L2CAP. Los mensajes L2CAP se pueden transmitir en un sólo paquete o fragmentados en varios paquetes *Baseband*. En cuyo caso se distingue entre el primer paquete (que lleva la cabecera L2CAP) y los sucesivos mediante el campo LLID de la cabecera del paquete *Baseband*. Este tipo de enlace es soportado normalmente por el transporte lógico ACL, aunque también puede serlo por el transporte SCO mediante el envío de paquetes DV.
  - SCO-S (*User Synchronous*): Se utiliza para la transmisión de un flujo síncrono de datos de usuario y se soporta sobre el transporte lógico SCO.
  - eSCO-S (*User Extended Synchronour*): Se utiliza para la transmisión de un flujo síncrono de datos de usuario y se soporta sobre el transporte lógico eSCO.

2.2.3.9. Creación de la *piconet*

Para construir una *piconet* se utilizan los procedimientos de *Inquiry* y *Paging*. El procedimiento *Inquiry* permite a un dispositivo descubrir la presencia de otros dispositivos en su entorno y obtener sus direcciones físicas. Para establecer conexiones con estos dispositivos, hay que utilizar el procedimiento de *Paging*. Cualquier dispositivo puede en principio ser maestro de una *piconet*. Por definición, el dispositivo que inicia la conexión es el que se convierte en maestro de la *piconet*, pudiendo iniciar conexiones con más de un dispositivo para formar *piconets* con varios esclavos. Una vez establecida la *piconet*, es posible iniciar un procedimiento para intercambiar el papel de los dispositivos (*Role Switch*), lo que permite sustituir al dispositivo maestro y conseguir que otro dispositivo tome el control de la *piconet*.

Estos procedimientos son necesarios porque para poder intercambiar información los dispositivos tienen primero que conseguir sincronizarse tanto a nivel de temporización de la trama, como a nivel de la secuencia de saltos de frecuencia.

Estados El diagrama de estados de un dispositivo *Bluetooth* se muestra en la Figura 2.6. El estado por defecto de un dispositivo *Bluetooth* se denomina *Standby*. En este estado el dispositivo está en modo de bajo consumo y no se encuentra conectado a ninguna *piconet*. Para iniciar una conexión hacia otro dispositivo, deberá conmutar al estado *Page*. Un dispositivo en *Standby* que acepta conexiones debe saltar periódicamente al estado *Page Scan*, en el que comprueba si algún otro dispositivo está intentando iniciar una conexión. Si es un dispositivo *descubrible* o visible para los demás, también saltará periódicamente al estado *Inquiry Scan*, en el que comprueba si algún dispositivo en la vecindad está emitiendo paquetes de búsqueda y en tal caso responde a dichos paquetes. Un dispositivo ejecutando el procedimiento de búsqueda se encuentra en el estado *Inquiry*.

Procedimiento de *Paging* Como se ha mencionado previamente, el procedimiento de *Paging* permite el establecimiento de conexiones entre diferentes módulos. Cuando un dispositivo inicia la conexión hacia otro dispositivo, deberá saltar al estado *Page*, en el que ejecuta la secuencia de *Paging*. Para que la conexión pueda ser posible es necesario que el dispositivo contrario, que actúa como esclavo, salte al estado *Page Scan* mientras el maestro está ejecutando la secuencia de *Paging*. Si el dispositivo esclavo detecta que está siendo direccionado por otro dispositivo se inicia un proceso mediante el cual el esclavo se sincroniza con el maestro y se establece la conexión.

Las acciones llevadas a cabo en el procedimiento de *paging* tienen como objetivo establecer la sincronización necesaria entre el maestro y el esclavo para llevar a cabo la comunicación. Como se detalla a continuación, para conseguir esta sincronización, en la primera fase del procedimiento el dispositivo que ejecuta el *Page Scan* escucha en la misma frecuencia durante un intervalo de tiempo relativamente largo, mientras que el que ejecuta la secuencia de *Page* cambia continuamente la frecuencia de transmisión.

Al realizar el *Page Scan*, el dispositivo esclavo permanece a la escucha en un determinado canal, e intenta detectar la transmisión de un paquete ID con su código de acceso (DAC) por parte de otros dispositivos. En este estado, el dispositivo cambia de frecuencia

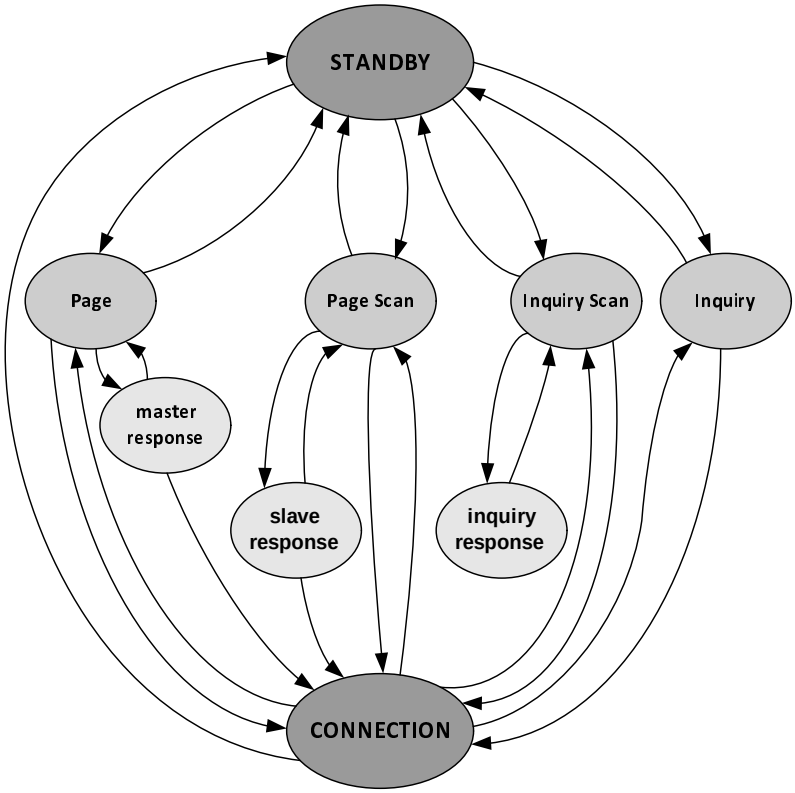


Figura 2.6: Diagrama de estados del *Link Controller*

cada 1,28 s, y emplea una secuencia de saltos especial (*page scan hopping sequence*) que sólo utiliza un subconjunto de 32 de los 79 canales radio disponibles.

Como el dispositivo maestro no conoce la frecuencia en la que está sintonizado el esclavo, intentará encontrarlo transmitiendo de forma repetida el DAC del esclavo en diferentes frecuencias, realizando un barrido por todas las posibles cambiando rápidamente de frecuencia. Dado que el paquete ID es suficientemente corto (68 bits), se puede transmitir dos veces en cada time-slot a diferentes frecuencias, de forma que se comprueban dos frecuencias por *time-slot*. Sin embargo, como después de enviar estos códigos de acceso el maestro tiene que esperar a la respuesta del esclavo en el siguiente time-slot y no puede transmitir en él, serían necesarios 16 *time-slots* (10 ms) para testear 16 frecuencias.

La secuencia de saltos utilizada por el dispositivo esclavo depende de su dirección, y la frecuencia seleccionada en un momento dado (el punto de la secuencia de saltos en el que se encuentra) la determina el reloj de dicho dispositivo. Para acelerar en lo posible el establecimiento de la conexión, el maestro realiza una estimación del reloj del esclavo que le permite determinar qué frecuencias son las que más probablemente estén siendo utilizadas por este. De esta forma el maestro genera una secuencia con las 16 frecuencias más próximas (dentro de la secuencia de saltos) a la estimada (secuencia A) y otra con las 16 frecuencias más alejadas (secuencia B). Esta secuencia de saltos empleada por el maestro se denomina *page hopping secuence*.

Cuando un dispositivo habilita la recepción de peticiones de conexión debe entrar periódicamente en el estado *Page Scan* (cada $T_{page-scan}$). El tiempo que permanece el

dispositivo en el estado *Page Scan* se denomina *page scan window* ($T_{wpage-scan}$), y debe ser suficiente para que el dispositivo maestro ejecute al menos 16 saltos de frecuencia (10 ms), aunque el valor por defecto es algo mayor (11,25 ms). Como el dispositivo maestro no conoce cuándo entra el esclavo en modo *Page Scan*, el barrido de frecuencias debe repetirse múltiples veces para asegurar que el esclavo detecte la solicitud de conexión. En el estado de *page* el maestro prueba N_{page} veces con las frecuencias de la secuencia A, y luego N_{page} veces con la secuencia B, repitiendo el proceso hasta que obtenga una respuesta del esclavo o expire un temporizador. Como es lógico, la configuración recomendada para N_{page} depende del intervalo $T_{wpage-scan}$ y del tipo de *scan* realizado por el dispositivo esclavo, que el maestro obtiene en el proceso de *inquiry*. Si éste no se ha realizado previamente, y no se conoce la configuración del esclavo, entonces $N_{page} = 256$

Cuando existe una conexión SCO o eSCO, la transmisión de paquetes síncronos tiene prioridad sobre los procedimientos de *Page* y *Page Scan*. En el caso de un dispositivo que realiza *page scan* la norma especifica que debe aumentar la ventana $T_{wpage-scan}$ para asegurar que consigue sincronizarse con un dispositivo que se encuentre realizando un *Page*. En el caso de que sea este último el que tiene conexiones SCO, el número de *slots* disponibles para realizar el barrido de frecuencias se reduce y la norma especifica que se debe aumentar el parámetro N_{page}.

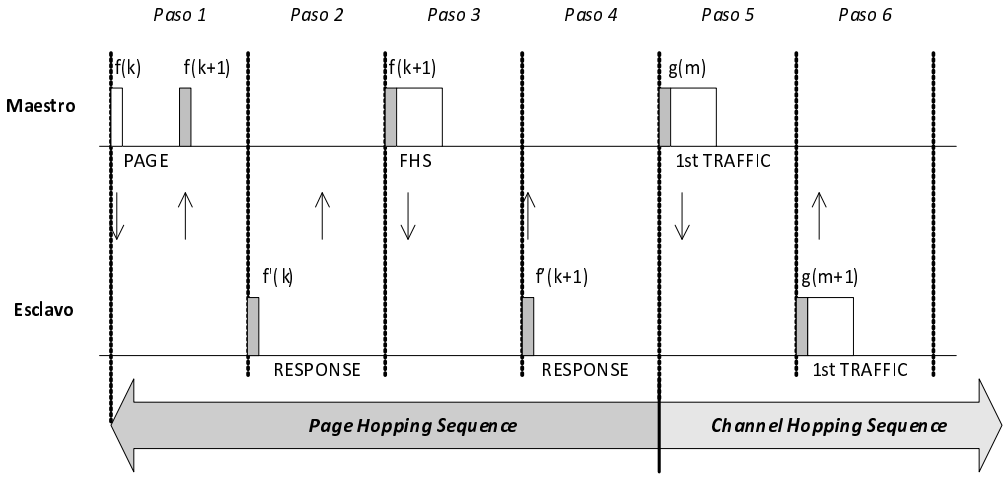
La detección de su código de acceso (DAC) indica al esclavo que realiza *Page Scan* que hay algún dispositivo intentando conectarse. Una vez que esto ocurre, ya existe cierta sincronización entre el esclavo y el maestro, puesto que ambos están utilizando la misma frecuencia. A partir de este momento el esclavo entra en el estado *Page Response* e inicia una secuencia de acciones que permite la completa sincronización con su maestro. Para que esto pueda ocurrir, el esclavo necesita conocer la dirección BD-ADDR del maestro (de la cual depende la secuencia de saltos de frecuencia que constituye el canal de la *piconet*, *basic channel hopping sequence*) y el reloj del mismo. El procedimiento por el que maestro y esclavo intercambian dicha información se realiza utilizando de nuevo la secuencia de saltos de frecuencia y el código de acceso del esclavo, ya que los parámetros del maestro (dependen de BD-ADDR) todavía no son conocidos por el esclavo. En los mensajes dirigidos del esclavo al maestro se utiliza la secuencia de saltos *page hopping sequence* mientras que los mensajes dirigidos del esclavo al maestro utilizan la secuencia *page response sequence*, que son similares pero calculadas de diferente forma.

El procedimiento completo se muestra en la figura 2.7 y consta de los siguientes pasos:

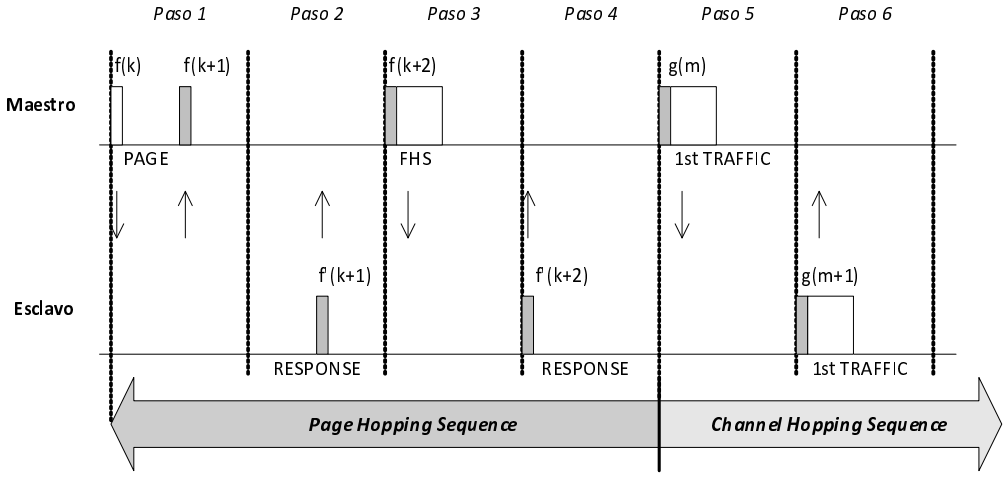
- El esclavo responde al mensaje de paging recibido, reenviando 625 μs después su código de acceso. La frecuencia utilizada para responder pertenece a la secuencia de saltos *page response*.

- El maestro envía al esclavo un paquete FHS con información de su BD-ADDR y su reloj. El código de acceso que utiliza es el del esclavo, y emplea la secuencia de saltos *page hopping sequence*. La detección del código de acceso mediante un correlador es utilizada por el esclavo para sincronizarse con el maestro. Además el paquete FHS indica al esclavo la dirección LT-ADDR local que se le asigna para su participación en la *piconet*.

- El dispositivo esclavo reconoce el paquete FHS enviado por el maestro en el siguiente time-slot.
- A partir de ese momento el maestro puede transmitir paquetes al esclavo. Para la comunicación se utiliza el código de acceso del canal y la secuencia de saltos de frecuencia del canal (dependientes ambos de la dirección del maestro). El control del acceso al canal es realizado por el maestro, que debe direccionar al esclavo (mediante un paquete con información o un paquete de POLL) para que este pueda acceder al canal.



(a) Secuencia en la que el esclavo responde al primer paquete de page



(b) Secuencia en la que el esclavo responde al segundo paquete de page

Figura 2.7: Cronograma de la etapa final del procedimiento de establecimiento de conexión

Hay dos variantes del procedimiento, tal y como se muestra en la figura 2.7, dependiendo de si el esclavo responde al primer o al segundo paquete ID enviado por el maestro. La respuesta del esclavo tiene lugar un *slot* después de haber detectado el paquete ID detectado, por lo que puede producirse al principio o en medio del *slot* de respuesta. En cualquiera de los casos la temporización de los *slots* queda determinada por el paquete FHS enviado por el maestro, que se transmite siempre transcurridos 1,25 ms desde el

comienzo del primero de los dos paquete ID enviados antes de la respuesta del esclavo, fijando la temporización de la conexión.

Una vez establecido el canal físico, los dispositivos pasan al estado *connected* y se intercambiarán por él una serie de paquetes de control, generados y procesados por las entidades *Link Manager* de ambos dispositivos, que determinarán más parámetros de configuración de la conexión

Procedimiento de *Inquiry* El procedimiento de *Inquiry* permite a un dispositivo obtener las direcciones físicas (BD-ADDR) de sus dispositivos vecinos, lo que le permitirá posteriormente iniciar conexiones hacia dichos dispositivos. El procedimiento de *Inquiry* es muy parecido al de *Page*, con la diferencia que al no ser necesario que los dispositivos se sincronicen, no se requiere la negociación final.

De la misma forma que en el caso anterior, los dispositivos que son *descubribles* entran de forma periódica en el estado *Inquiry Scan*, en el que examinan una determinada frecuencia en busca de mensajes de *Inquiry*. La frecuencia que examina un dispositivo cambia cada 1,28 s y pertenece a una secuencia de saltos especial denominada *inquiry hopping sequence*, que de nuevo consta de 32 frecuencias diferentes. En este modo, se espera la recepción de mensajes de *Inquiry*, que consisten únicamente en un paquete tipo ID con un código de acceso especial denominado IAC, que es detectado mediante un correlador.

El mensaje de *Inquiry* es transmitido múltiples veces por el dispositivo que realiza el procedimiento de *Inquiry* a diferentes frecuencias, ya que no se conoce las frecuencias a las que están escuchando los dispositivos vecinos. Como además tampoco se conocen los instantes de tiempo en los que estos dispositivos pasan al estado *Inquiry Scan*, el barrido de frecuencias se repite un número de veces $N_{inquiry}$ suficiente para asegurar que el mensaje de *Inquiry* es detectado por algún dispositivo.

La duración del estado *Inquiry Scan* se denomina ventana de *Inquiry* ($T_{winquiry}$), y debe ser lo suficientemente larga como para que el dispositivo que realiza el *Inquiry* pueda realizar un barrido de 16 frecuencias. De la misma forma que en el procedimiento de *Page*, el dispositivo que realiza el *Inquiry* transmite el código de acceso a dos frecuencias diferentes en el mismo time-slot y deja un time-slot para la respuesta, por lo que se pueden barrer 16 frecuencias en 10 ms. El conjunto de 32 frecuencias que pueden formar parte de la secuencia de saltos es separado de nuevo en dos grupos de 16 frecuencias cada uno (secuencias A y B). El primer grupo es barrido $N_{inquiry}$ veces y luego se escanea el grupo B otras tantas veces. Como el periodo con el que un dispositivo salta al estado *Inquiry Scan* $T_{inquiry-scan}$ puede ser de hasta $2,56s$, $N_{inquiry}$ se fija a por defecto a 256 para asegurar un funcionamiento correcto. Además, para recoger el mayor número de respuestas, este proceso se repite hasta 3 veces, por lo que el procedimiento de *Inquiry* puede durar hasta $10,24s$, excepto en el caso de que el dispositivo que realiza el proceso reciba un número suficiente de respuestas como para abortar el proceso. Si existen conexiones SCO en el maestro o en el esclavo, los parámetros $T_{winquiry}$ y $N_{inquiry}$ deben ser aumentados respectivamente, ya que dichos paquetes son enviados de forma periódica y tienen prioridad sobre el procedimiento de *Inquiry* e *Inquiry Scan*.

El código de acceso que se utiliza en el proceso de *Inquiry* (IAG) puede ser un código

dedicado (DIAC) o un código general (GIAG). El código dedicado se utiliza para buscar un tipo concreto de dispositivo, ya que sólo responden al proceso de *Inquiry* aquellos dispositivos que tienen el mismo DIAC. El DIAC depende generalmente de un parámetro del dispositivo, generalmente configurable, que indica la aplicación a la que está destinado (*Class of device*). El código de acceso genérico (GIAC) es común para todos los dispositivos *Bluetooth* del mundo.

Cuando un dispositivo que está realizando un *Inquiry Scan* detecta el código de acceso correspondiente (GIAC o DIAC), envía como respuesta en el siguiente time-slot, $0,625ms$ después, un paquete FHS que contiene los parámetros del dispositivo (BD-ADDR, reloj). El dispositivo que realiza el *Inquiry* lee el mensaje de respuesta, pero no detiene el proceso, sino que continúa enviando mensajes de *Inquiry* para detectar otros dispositivos. La información de reloj enviada en el paquete FHS permite al dispositivo que realiza la búsqueda estimar el reloj de sus dispositivos vecinos, permitiendo mejorar la eficiencia del procedimiento de conexión con ellos en caso de que se realice. El paquete FHS permite también conocer alguna información adicional sobre la configuración y comportamiento del dispositivo encontrado. Algunos dispositivos soportan el modo de respuesta extendida al *inquiry* (EIR, *Extended Inquiry Response*) introducido a partir de la versión 2.1, en cuyo caso transmiten un paquete adicional de datos ACL (DM o DH de 1, 3 o 5 *slots*) 1,25 ms después del paquete FHS, aportando información sobre los servicios soportados por el dispositivo.

El hecho de que varios dispositivos puedan detectar el código de acceso y responder de forma simultánea provoca que se puedan producir colisiones. Aunque esto no debe ocurrir con mucha probabilidad, puesto que los dispositivos no están en principio sincronizados y deberían por tanto encontrarse en puntos diferentes de la secuencia de frecuencias, es necesario evitar que se produzcan colisiones de forma repetitiva. Para ello se introduce un mecanismo de backoff que establece que un dispositivo que detecte el mensaje de *Inquiry* mientras está ejecutando un *Inquiry Scan* no volverá a entrar en dicho estado hasta transcurrido un número aleatorio de *slots* comprendido entre 0 y MAX_RAND (por defecto 1023), además de saltar a la siguiente frecuencia dentro de la secuencia. Transcurrido ese tiempo vuelve al estado *Inquiry Scan* y responde con un nuevo paquete FHS cuando vuelve a detectar el código de acceso correspondiente al *Inquiry*. De esta manera, cada dispositivo responde de media unas 4 veces durante una ventana de 1,28 segundos, pero lo hace a diferentes frecuencias y en instantes de tiempo aleatorios.

Otra posibilidad de que se produzcan colisiones es que más de un dispositivo inicie simultáneamente el proceso de *Inquiry*. Para evitar esto, especialmente en el caso de que el proceso deba realizarse de forma repetitiva, es aconsejable introducir a nivel de aplicación una componente aleatoria en los tiempos que evite la aparición de cualquier forma de periodicidad.

Secuencias de salto de frecuencia Como se ha mencionado en los apartados anteriores, existen diferentes secuencias de salto de frecuencia en función de la actividad que se está realizando:

- En el estado *Connected* se sigue la secuencia de saltos del canal, *basic channel hop-*

ping sequence impuesta por el maestro, que tiene un periodo de repetición largo y utiliza hasta 79 frecuencias distintas, cambiando de frecuencia cada time-slot. Si se está transmitiendo un paquete de más de un time-slot de duración la frecuencia se mantiene hasta la finalización del paquete. El tiempo desde la finalización de la transmisión de un paquete en un time-slot y el comienzo del siguiente time-slot es suficiente como para saltar de frecuencia.

- Además, en el estado *Connected*, en caso de que maestro y esclavo soporten AFH, una vez establecida la conexión las entidades *Link Manager* de ambos dispositivos pueden acordar utilizar una secuencia de saltos modificada (*Adapted channel hopping sequence*), en la que algunos canales radio se pueden eliminar cuando se detecta que las condiciones de transmisión en ellos no son adecuadas.

- En el estado *Page Scan*, el dispositivo esclavo utiliza la secuencia *page hopping sequence* para determinar la frecuencia en la que recibe, cambiando cada 1,28 s. Una vez iniciado el diálogo con el maestro, esta secuencia se utiliza en las transacciones entre el maestro y el esclavo, cambiando de frecuencia en cada time-slot. La secuencia page depende de la dirección del propio dispositivo y está limitada a sólo 32 posibles frecuencias.

- Para las transacciones entre el esclavo y el maestro durante la negociación previa al establecimiento de conexión, se utiliza la secuencia *page response hopping sequence*, que también depende del esclavo y consta de 32 posibles frecuencias diferentes.

- En el procedimiento de *Inquiry*, y de forma similar al procedimiento de *Page*, se utilizan las secuencias *inquiry hopping sequence* e *inquiry response hopping sequence*.

2.2.4. Gestión de la red y modos de bajo consumo

Los dispositivos integrados en una *piconet* se encuentran en el estado *connection*, al que se entra, como esclavo o como maestro, después del procedimiento de *paging*. El estado de conexión comienza con un paquete POLL enviado por el maestro para verificar el cambio de la sincronización del maestro y la secuencia de saltos del canal. El esclavo puede contestar con cualquier tipo de paquete, pero si por algún motivo no se consiguiese establecer la comunicación, ambos vuelven a los estados de *Page*/*Page Scan*. Si por el contrario la comunicación tiene éxito, el canal físico de la *piconet* queda establecido y se crea un transporte lógico ACL por el cual las entidades *Link Manager* de ambos dispositivos se pueden intercambiar mensajes de control, negociando algunos parámetros de configuración de la conexión y/o la creación de transportes y enlaces lógicos adicionales (por ejemplo SCO), así como el cambio de rol o de modo de operación.

Durante el estado de conexión un módulo esclavo puede encontrarse en varios modos de operación o subestados (ver figura 2.8). El modo activo es el modo por defecto, en el que un dispositivo está activamente participando en la *piconet*. Los modos *sniff* y *hold* son modos de bajo consumo en los que el dispositivo esclavo no está presente en el canal físico en todo momento, mientras que el modo *park* es un modo de bajo consumo más profundo en el que el dispositivo abandona temporalmente la *piconet* (perdiendo incluso su dirección

LT-ADDR) pero se mantiene sincronizado con ésta, de forma que puede reintegrarse de una forma más rápida y eficiente energéticamente. El dispositivo maestro estará siempre en modo activo.

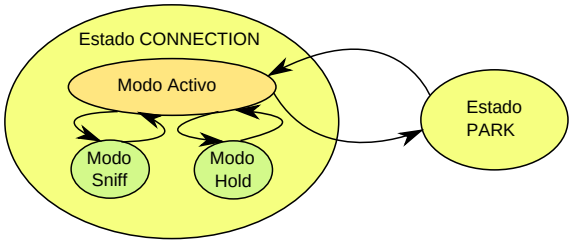


Figura 2.8: Temporización de la transmisión maestro esclavo (caso básico)

2.2.4.1. Modo activo

Como ya se ha mencionado anteriormente, el control de una *piconet* corresponde al dispositivo maestro, siendo éste el que controla la temporización de la trama y los saltos de secuencia, por lo que el resto de dispositivos que integran la *piconet* deben permanecer sincronizados con él. El maestro podrá iniciar sus transacciones en los *slots* pares, mientras que los esclavos podrán hacerlo en los impares, siempre y cuando el maestro les haya direccionado en el *slot* anterior. La planificación (*scheduling*) del acceso a los recursos del canal es realizada por el dispositivo maestro, y ningún dispositivo esclavo podrá transmitir en un determinado *slot* si no es autorizado previamente por él.

La secuencia pseudoaleatoria de saltos de frecuencia y el código de acceso al canal utilizados en la *piconet* son fijados también por el maestro, y son derivados de su dirección física (BD-ADDR). Siguiendo dicha secuencia de saltos, la frecuencia de transmisión se cambia en cada *slot* (figura 2.9). Los paquetes *multislot* se transmiten a una única frecuencia, que corresponde con la del *slot* en el que empezó la transmisión, pero la siguiente frecuencia de transmisión se selecciona según el número de *slots* transcurridos, de manera que se mantiene la coherencia temporal independientemente del tipo de paquete enviado (figura 2.10). Esto también ocurre cuando no se realizan transmisiones durante algunos *slots*. En el caso de que se utilice la secuencia de saltos adaptada AFH, los paquetes descendentes (de maestro a esclavo) y ascendente (de esclavo a maestro) consecutivos se transmiten a la misma frecuencia, realizando el salto de frecuencia cada dos paquetes, si bien en el cálculo de la nueva frecuencia se tiene en cuenta igualmente el número de *slots* transcurridos.

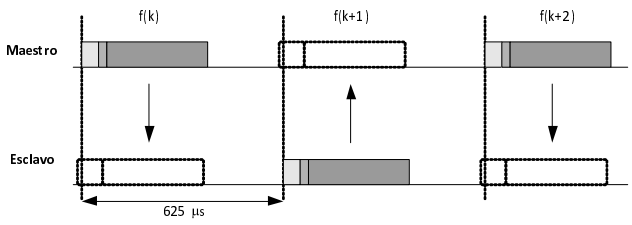


Figura 2.9: Temporización de la transmisión maestro esclavo (caso básico)

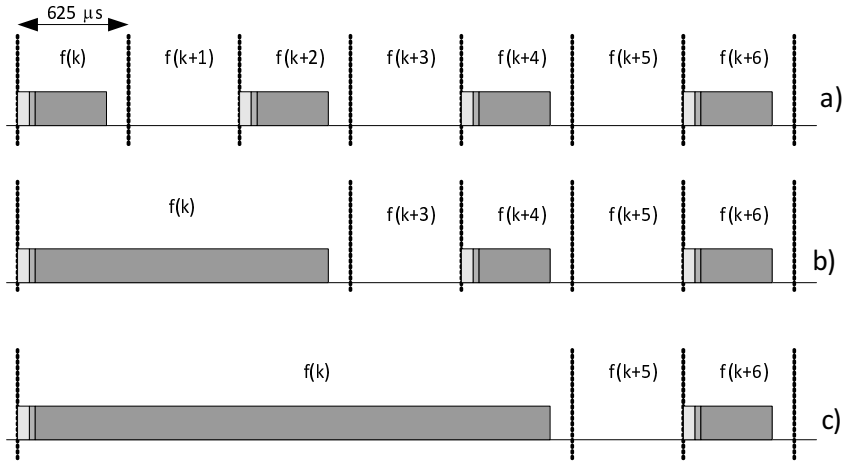


Figura 2.10: Temporización de la transmisión maestro esclavo con paquetes *multislot*

El maestro planifica las transmisiones hacia y desde el esclavo en función de la demanda de tráfico. Los esclavos activos, que como ya se ha mencionado pueden ser hasta 7, deben escuchar en los *slots* maestro-esclavo a la espera de ser direccionados, y en caso de no serlo pueden quedar dormidos hasta la siguiente posible transmisión por parte del maestro, cuyo posible comienzo vendrá determinado por el tipo de paquete que se esté enviando (1, 3 o 5 *slots*). Aunque no haya tráfico que transmitir, el maestro debe asegurarse de enviar paquetes de forma periódica para refrescar la temporización del canal físico y mantener a los esclavoc sincronizados. Para esto los esclavos sólo necesitan recibir el código de acceso al canal, por lo que cualquier tipo de paquete es válido para ello, incluido aquellos paquetes dirigidos a otros esclavos o con errores de transmisión. Además, el maestro debe garantizar a los esclavos que van a ser direccionados, y por tanto que tendrán la oportunidad de transmitir información, al menos cada cierto intervalo tiempo T_{poll}. Este periodo de sondeo es un parámetro de calidad de servicio que determina la latencia y la tasa de envío de cada dispositivo esclavo, y es establecido mediante mensajes de control de la capa LMP. Para direccionar a los esclavos, el maestro utiliza los identificadores locales LT-ADDR, que se asignan a los esclavos durante el procedimiento de apertura de conexión.

Además del transporte lógico ACL, e intercambiando mensajes de control mediante el mismo, las entidades *Link Manager* pueden establecer transportes lógicos SCO y eSCO, que tienen un conportamiento diferente al estar destinados a tráfico síncrono. En el caso del SCO, el maestro debe direccionar al esclavo respetando el periodo T_{SCO}, y el esclavo debe transmitir en el *slot* previsto siempre que sea direccionado por el maestro. En el caso de los transportes eSCO, que admiten retransmisión de paquetes, además de la periodicidad con la que se producen las comunicaciones entre maestro y esclavo (T_{eSCO}), se negocia una ventana de transmisión D_{eSCO}, y se utilizan dos reglas de sondeo. El maestro debe transmitir hacia el esclavo respetando la temporización marcada por (T_{eSCO}) utilizando el/los primero(s) *slot* de la ventana, y el esclavo debe responder una vez direccionado por el maestro (de forma similar a un enlace SCO). Una vez realizada esta comunicación, se sigue un esquema similar al ACL durante el resto de la ventana negociada para la retransmisión: el maestro podrá en cualquier *slot* impar enviar información o sondear al esclavo y éste

podrá responder en el siguiente *slot* si el maestro le ha direccionado.

2.2.4.2. Modo *sniff*

El modo *sniff* se utiliza para reducir el ciclo de trabajo de la actividad de escucha realizada por el esclavo, permitiendo reducir su consumo. Esto se consigue restringiendo los *slots* en los que el maestro puede comenzar una transmisión hacia un esclavo determinado, lo que permite al esclavo no tener activa la recepción continuamente.

La entrada en modo *sniff* es negociada por los *Link Manager* de los dispositivos mediante el intercambio de mensajes de control que determinan los parámetros que se van a utilizar. Entre estos parámetros se encuentran el *offset* (D_{sniff}) y el intervalo de repetición (T_{sniff}), que determinan cuando comienza el intervalo de *sniff* y cada cuanto se repite. El maestro puede transmitir al comienzo de dicho intervalo de *sniff*, también denominado punto de anclaje (*sniff anchor point*), y por tanto el esclavo debe activar el receptor al comienzo de dicho intervalo.

La duración del intervalo de escucha depende de otros dos parámetros, $N_{attempt}$ y $N_{timeout}$, que también se establecen en la negociación entre el maestro y el esclavo. El esclavo permanece escuchando posibles paquetes provinientes del maestro siempre y cuando hayan transcurrido menos de $N_{attempt}$ *slots* desde el punto de anclaje, o menos de $N_{timeout}$ *slots* desde la recepción de un paquete con su LT-ADDR desde el maestro o desde el envío de un paquete propio hacia el maestro. El doble mecanismo se establece para que se pueda compaginar una actividad reducida del esclavo cuando el maestro no envía nada con una buena velocidad de respuesta cuando sí hay tráfico que enviar. La figura 2.11 muestra un ejemplo de la temporización del modo *sniff* con $N_{attempt} = 2$, $N_{timeout} = 1$ y $T_{sniff} = 6slot$

La extensión del periodo de actividad de forma reiterada por envío de paquetes por parte del maestro puede hacer que un intervalo de *sniff* se extienda hasta el comienzo del siguiente, en cuyo caso no debe haber solape, y se tomará el nuevo punto de anclaje como referencia para determinar el fin del la escucha del esclavo.

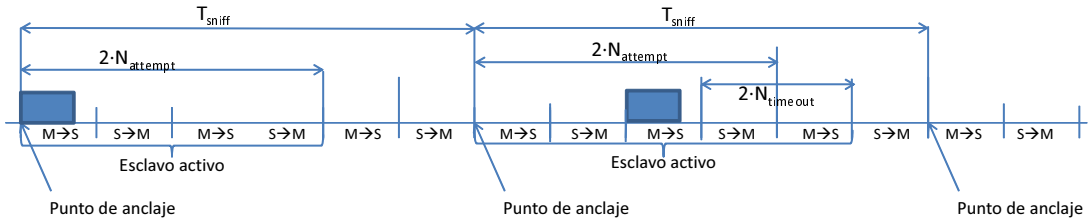


Figura 2.11: Temporización del modo *sniff*

A partir de la versión 2.1 de las especificaciones se introdujo un nuevo mecanismo para mejorar el consumo del esclavo en modo *sniff* denominado *sniff subrating*, que permite que el esclavo permanezca inactivo durante un número reducido de puntos de anclaje. Un dispositivo esclavo en modo *sniff* con este mecanismo habilitado pasa al subestado *sniff subrate* si transcurrido un número configurable de puntos de anclaje no ha recibido ningún paquete del maestro. Una vez en el subestado *sniff subrate*, el esclavo sólo se despierta para escuchar cada cierto número (también configurable) de puntos de anclaje. Si cuando

está despierto recibe algún paquete del maestro dirigido hacia él, entonces debe retornar al modo *sniff* y volver a despertarse en todos los puntos de anclaje.

2.2.4.3. Modo *hold*

Durante el estado de conexión, un dispositivo esclavo puede entrar en este modo en el que temporalmente no soporta paquetes ACL. De esta manera el esclavo ahorra recursos que puede utilizar en otras operaciones como *Paging* o *Inquiry* o atender a otra *piconet*. Por otra parte, si nada de esto es necesario, el módulo puede entrar en modo de bajo consumo. No obstante, durante el modo *hold* el esclavo mantiene su dirección LT-ADDR activa.

Antes de entrar en este modo el maestro y el esclavo acuerdan el tiempo que el esclavo va a estar en modo *hold*, inicializando un timer con dicho valor. Cuando el timer expira, el dispositivo deberá sincronizarse con el tráfico en el canal y esperar las subsiguientes instrucciones del maestro.

2.2.4.4. Modo *park*

En este modo un esclavo no participa en la *piconet* pero se mantiene sincronizado al canal mediante una pequeña actividad que no requiere mucho consumo. Mientras permanece aparcado, el esclavo libera su identificador de miembro activo de la *piconet*, LT-ADDR, recibiendo en su lugar dos nuevas direcciones de 8 bits cada una que serán utilizadas en este modo:

- PM-ADDR: Parked Member Address
- AR-ADDR: Access Request Address

La dirección de miembro aparcado (PM-ADDR) es única para cada dispositivo aparcado y lo distingue de los demás dispositivos aparcados, siendo principalmente utilizada por el maestro para activarlo. No obstante, esta dirección no es imprescindible, puesto que este proceso se puede realizar también con la dirección global de dispositivo *Bluetooth* (BD-ADDR), lo que se indica cuando utilizando el valor especial PM-ADDR=0. Por otra parte, la dirección AR-ADDR es usada por el esclavo cuando éste inicia el proceso para abandonar el modo *park*.

Debido a la pérdida del identificador LT-ADDR, todos los paquetes del maestro hacia los esclavos aparcados deben ser *broadcast*. Un esclavo en modo *park* se despierta regularmente y escucha el canal para resincronizarse y buscar paquetes *broadcast*. Para facilitar esta sincronización, el maestro soporta un canal piloto (beacon channel) cuya configuración se comunica al esclavo al ser aparcado. Si dicha configuración cambia, la configuración de los esclavos se actualiza mediante paquetes *broadcast*.

Además de reducir el consumo de los módulos que se encuentran aparcados, este modo posibilita la creación de *piconets* formadas por un mayor número de esclavos sincronizados con el maestro. La limitación de reducir a 7 esclavos el tamaño de la *piconet*, debida al tamaño del identificador LT-ADDR, puede ser sorteada si los nodos conmutan entre los

estados *active* (*connected*) y *park*. El número de esclavos se amplía hasta 255 si se utiliza la dirección PM-ADDR, pero pueden ser virtualmente ilimitados si se usa la BD-ADDR para desaparcarlos.

Canal piloto (*Beacon channel*) Con el objeto de soportar uno o más esclavos aparcados, el dispositivo maestro debe establecer un canal piloto consistente en un tren de paquetes piloto enviados en *slots* equidistantes y repetidos de forma periódica, según se observa en la figura 2.12. El número de paquetes piloto en un tren N_B y el intervalo de repetición de dicho tren T_B son parámetros configurables que se negocian entre las entidades *Link Manager* junto con el *offset* D_B, que indica el instante de comienzo, y la separación entre paquetes pilotos, Δ_B (ver figura 2.12).

Las funciones del canal piloto son:

- Transmisión de paquetes del maestro hacia el esclavo que éstos utilizan para sincronizarse. Cualquier tipo de paquete es válido, si no hay información que transmitir se envía un paquete NULL.
- Enviar mensajes hacia los esclavos para cambiar parámetros del propio canal piloto
- Enviar mensajes generales de *broadcast* a los esclavos aparcados.
- Desaparcar a uno o más esclavos.

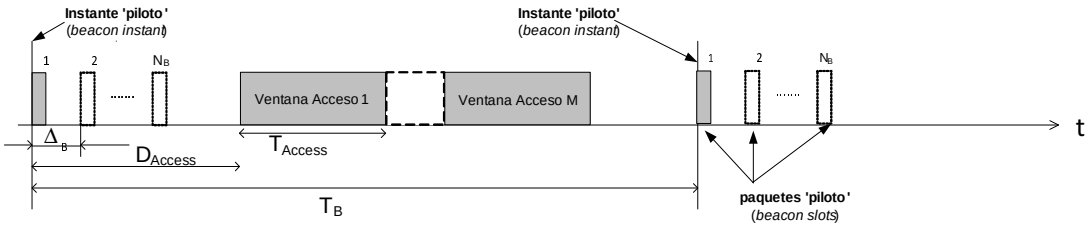


Figura 2.12: Temporización de los trenes de paquetes del canal piloto y ventanas de acceso

Ventana de acceso. Además de los *slots* pilotos, se define una ventana de acceso en la cual un esclavo en modo *park* puede enviar peticiones para salir de dicho modo. Esta ventana se puede repetir un máximo de M_{access} veces después del comienzo del tren de paquetes pilotos (*beacon instant*) y ha de estar separada de ésta un mínimo de D_{access} *slots*. La duración de la ventana es T_{access} (Ver Figura 2.12).

Durante estas ventanas de acceso, el maestro realiza un sondeo de los esclavos aparcados para que estos indiquen si quieren ser desaparcados, respetando siempre la estructura TDD. El funcionamiento de esta secuencia se muestra en la figura 2.13, en la que se observa que las transmisiones maestro-esclavo, realizadas mediante paquetes *broadcast*, son alternadas con transmisiones esclavo-maestro. Los *slots* destinados a los esclavos, son a su vez divididos en dos mitades de 312,5 μs en las que los dispositivos aparcados pueden enviar un paquete ID. El *semi-slot* que corresponde a cada esclavo aparcado depende de

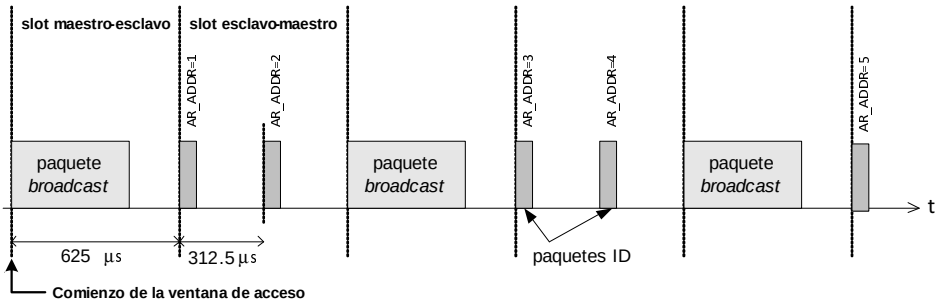


Figura 2.13: Secuencia de sondeo en la ventana de acceso

su dirección AR-ADDR. Cada esclavo debe enviar únicamente por el *semi-slot* que le corresponde, siempre y cuando se haya recibido el paquete *broadcast* del maestro en el *slot* anterior. De esta forma, el maestro lleva la iniciativa en el sondeo de los esclavos aparcados, pudiendo inhabilitar la transmisión de los esclavos aparcados cuando el *slot* se necesite para tráfico más prioritario o se encuentre reservado. Además, el maestro puede desactivar la ventana de acceso correspondiente a un tren concreto mediante la señalización enviada en los paquetes piloto, evitando así que los esclavos aparcados se despierten innecesariamente cuando el maestro no tiene intención de sondearlos. El número de *slots* de la ventana de acceso $N_{acc-slot}$ es indicado por el maestro a los esclavos al aparcarlos.

Sincronización de los esclavos en modo *park*. Los esclavos en modo *park* duermen durante la mayor parte del tiempo, pero se despiertan periódicamente para resincronizarse con el canal. Cualquier paquete intercambiado en el canal puede ser utilizado para dicha sincronización, por lo que en general aprovechan la obligación que tiene el maestro de transmitir en los *slots* pilotos. Así el esclavo tiene N_B oportunidades de sincronizarse durante un intervalo de repetición entre dos ráfagas de *slots* piloto (T_B). No obstante, no es necesario que el esclavo se active en el comienzo de cada ráfaga, sino que se puede definir un intervalo durante el cual el esclavo permanece inactivo y que puede ser mayor que T_B, aunque siendo siempre un múltiplo de este. Este intervalo se denomina *sleep window*.

Abandono del modo *park* por iniciativa del maestro. Un maestro puede desaparcar a un esclavo enviándole un comando LMP *unpark*. Este mensaje es enviado en un paquete *broadcast* en un *slot* piloto. Para identificar al esclavo se usa la PM-ADDR o la dirección completa del esclavo (BD-ADDR). El mensaje también incluye el identificador LT-ADDR que el esclavo utilizará al abandonar el modo *park* y agregarse a la *piconet*. Este mensaje puede contener varias direcciones de esclavos con sus correspondientes LT-ADDR, de forma que se puede desaparcar a varios esclavos a la vez.

Después de abandonar el modo *park* el esclavo sigue escuchando al maestro hasta ser direccionado mediante su LT-ADDR. El primer paquete enviado por el maestro debe ser un paquete POLL y la respuesta a éste por parte del esclavo confirmará que ha salido del modo *park*. Si el esclavo no recibe el paquete POLL volverá al modo *park*, y si el maestro no recibe respuesta al paquete POLL pondrá al esclavo de nuevo en modo *park*. Una vez que el esclavo está activo, el maestro decidirá en que modo continua (activo, *sniff* ...).

Abandono del modo *park* por iniciativa del esclavo. Como se ha detallado previamente, el esclavo puede solicitar el acceso a la *piconet* transmitiendo un paquete durante la ventana de acceso en su medio *slot* correspondiente. El mensaje que envía el esclavo es un paquete ID que contiene el código de acceso al dispositivo (DAC) del maestro.

Después de enviar una petición de acceso, el esclavo se mantiene a la espera de recibir un mensaje *unpark* del maestro. Si esto no sucede, el esclavo repite el envío de una petición de acceso en la siguiente ventana de acceso. Tras la última ventana de acceso, el esclavo esperará un tiempo adicional N_{poll} la llegada del mensaje *unpark*. Si esto no ocurre, el esclavo vuelve a dormirse e intenta este proceso tras la próxima ráfaga de *slots* pilotos (*beacon instant*). Si se recibe el mensaje *unpark* el proceso es el mismo que en el apartado anterior.

2.2.4.5. Consumo de los dispositivos.

Aparte de la optimización de receptor y transmisor para un bajo consumo hay dos factores que condicionan el consumo. El primero es el número de *slots* que ocupa el paquete y que se indica en el campo TYPE. Un esclavo no direccionado en el primer *slot* puede dormirse el resto de los *slots* que ocupa el paquete.

El segundo factor es el modo en el que se encuentre el esclavo. Una posible clasificación de los modos en orden decreciente de consumo sería: *sniff*, *hold* y por último el modo *park* con el menor consumo. El modo que más consume es el modo activo. En todos los modos mencionados, el dispositivo está en el estado *Connected*. El estado *Standby* es realmente el que menos consume, pero en tal estado el módulo no está conectado, y debe saltar periódicamente a los estados *Page Scan* e *Inquiry Scan* para atender las peticiones de conexión de otros dispositivos.

2.2.4.6. *Scatternet.*

El sistema de salto de frecuencia que incorpora *Bluetooth* permite que varias *piconets* puedan coexistir en el mismo área, ya que la secuencia de saltos de frecuencia depende del maestro que es único para cada *piconet*. Además, los paquetes intercambiados en el canal de una *piconet* están precedidos por el código de acceso al canal determinado por la dirección del maestro, que es fácilmente detectable mediante el uso de un correlador. Aunque las técnicas FH facilitan la operación de múltiples sistemas sin realizar una planificación de frecuencias, es evidente que mientras más *piconets* compartan el espacio radioeléctrico mayor será la probabilidad de que se produzcan colisiones y por tanto el rendimiento se degrada. No obstante, esta degradación de la capacidad del sistema es progresiva y no abrupta (*soft capacity*).

Cuando se produce esta situación de coexistencia de varias *piconets* en una misma zona, un dispositivo puede participar en dos o más de ellas simultáneamente mediante el uso de la multiplexación en el tiempo. Para participar en el canal correcto, debe utilizar la dirección del maestro y el *offset* correspondiente para obtener la fase adecuada. Un dispositivo puede actuar como esclavo en varias *piconets* simultáneamente pero sólo en una como maestro, ya que si fuera maestro de dos *piconets* simultáneamente éstas estarían

sincronizadas entre sí y utilizarían la misma secuencia de salto de modo que a todos los efectos, serían la misma *piconet*. Cuando existen esclavos participando en varias *piconets* se dice que se ha formado una *scatternet*.

Para la formación de la *scatternet* se utiliza el mismo procedimiento de *Paging* descrito para la formación de las *piconets*. Como el proceso de *Paging* siempre es iniciado por el maestro, puede ser necesario un intercambio de roles (que se explica más adelante) para obtener la configuración final deseada.

Para que un dispositivo participe en una *scatternet* debe utilizar multiplexación en el tiempo para ir cambiando de una *piconet* a otra. Para conseguirlo, los dispositivos pueden utilizar el modo *sniff* o mantenerse en una conexión ACL activa. Cuando el dispositivo es esclavo en estado *Connection*, puede elegir no escuchar en todos los *slots* del maestro. En este caso, la calidad de servicio en el enlace se degrada considerablemente si el esclavo no está presente lo suficiente como para coincidir el instante en que el maestro le está enviando los paquetes de *polling*. Del mismo modo, en la *piconet* en que el dispositivo es maestro, puede elegir no transmitir en todos sus *slots*. En este caso es muy importante cumplir el período de sondeo T_{poll} de una forma lo más exacta posible.

Los dispositivos en modo *sniff* pueden tener tiempo suficiente de conmutar a otras *piconets* entre *slots* de *sniff* consecutivos. Cuando el dispositivo es un esclavo en modo *sniff* y no existen suficientes *slots* libres, el dispositivo puede elegir no escuchar en todos los *slots* de transmisión del maestro de acuerdo a los parámetros número de intentos y *timeout*. El dispositivo maestro no está obligado a transmitir durante los *slots* de *sniff* y por tanto, tiene flexibilidad para participar en la *scatternet*.

Una unidad que participa en varias *piconets* tiene que usar en cada acceso la dirección del dispositivo maestro y el *offset* del reloj correspondientes a la *piconet* en la que quiera participar. Como los relojes de los maestros de diferentes *piconets* no están sincronizados, un esclavo que participe en dos *piconets* debe mantener dos *offsets* que, añadidos a su reloj nativo, generan los relojes de los dos maestros. Como la deriva de los relojes de ambos maestros es independiente, el esclavo debe actualizar regularmente los valores de *offset* para mantener la sincronización con los dos maestros.

Aunque la norma no establece una limitación sobre en cuantas *piconets* puede participar un dispositivo como esclavo, es evidente que en la práctica se producirán limitaciones debido a la necesidad de permanecer sincronizado con todas ellas. En la mayoría de dispositivos comerciales analizados durante el desarrollo de la tesis, se ha comprobado que normalmente no aceptan la conexión a más de dos *piconets*, y que en caso de aceptar la conexión a 3, ésta se comporta de forma muy inestable que rápidamente deriva en la pérdida de una de las tres conexiones.

2.2.4.7. Cambio de rol

Existen varias situaciones en las que se requiere una conmutación de este tipo, por ejemplo cuando un esclavo en una *piconet* quiere formar una nueva *piconet* en la que ha de actuar como maestro. Un posible caso sería el que un dispositivo inicia una conexión hacia otro que ya es maestro de una *piconet*. En tal caso el que inicia la conexión se convierte en maestro de una *piconet* formada por dos dispositivos, y su esclavo es maestro

de la *piconet* preexistente. Si sin embargo, el dispositivo iniciador quisiera unirse a la *piconet* inicial, en lugar de permanecer como maestro, debería utilizar el procedimiento de cambio de papeles (*Role Switching*). Finalmente, existe la posibilidad de que un dispositivo sea esclavo de una *piconet* y maestro de otra, formando una red jerárquica.

A nivel de la capa *Baseband* el cambio de papeles sigue básicamente dos etapas: En primer lugar se produce un cambio en la temporización TDD, pasando el nuevo maestro a transmitir en los *slots* en los que antes recibía, aunque manteniendo todavía los parámetros antiguos del canal. Una vez hecho esto negocian los nuevos parámetros del canal mediante el envío de un paquete FHS por parte del nuevo maestro, que debe ser reconocido por el antiguo. Este paquete contiene además el identificador LT-ADDR que utilizará el maestro antiguo en la nueva *piconet*, que no tiene por qué ser igual que el utilizado por el maestro antiguo. Una vez negociados los parámetros las dos unidades pueden ya utilizar el nuevo canal (*piconet switching*).

Es importante descatar que el procedimiento de *role switch* se lleva a cabo entre los dispositivos de forma bilateral. Si la *piconet* original tenía varios dispositivos esclavos, cuando el maestro realiza el *role switch* con uno de ellos el resto continúa como esclavos del maestro original, y este a su vez queda como esclavo del nuevo maestro, formandose una *scatternet*. Si se parte de la situación contraria, es decir, si un dispositivo es maestro de otro, que es a su vez maestro de uno o varios esclavos, puede integrarse en la *piconet* de su esclavo al realizar el *role-switch* con él. Escenarios en los que se quiera realizar la sustitución del maestro de una *piconet* con varios dispositivos por uno de sus esclavos, o fusionar dos *piconets* con varios dispositivos cada una son complejos y no están soportado por la especificación de *Bluetooth* actual.

Los esclavos que se encuentren aparcados y tengan que participar en un cambio de *piconet* deben ser desaparcados previamente. La encriptación debe ser también desactivada antes de iniciar el procedimiento de *Role Switching*. Además, el transporte lógico ACL se reiniciará, como si se hubiera creado una conexión nueva.

2.2.5. Nivel *Link Manager*

La entidad gestora de enlaces (*Link Manager*) es la responsable de la creación, modificación y liberación de enlaces lógicos y de los transportes lógicos asociados, así como de la actualización de parámetros relativos a los enlaces físicos entre dispositivos. Para esto es necesario la comunicación entre las entidades homólogas del maestro y del esclavo, que se lleva a cabo mediante el protocolo de gestión de enlace LMP.

El protocolo LMP consta de una serie de mensajes que deben ser transmitidos sobre el enlace lógico ACL-C en el transporte lógico ACL por defecto que existe entre dos dispositivos conectados. Los mensajes LMP deben ser interpretados y ejecutados por los gestores de enlace y no deben transmitirse a capas superiores. Los mensajes intercambiados entre el dispositivo maestro y esclavo son aplicables únicamente al enlace físico entre ellos y a sus enlaces lógicos y transportes lógicos asociados.

2.2.6. Interfaz *Host-Controller* (HCI)

Con este nombre se denomina al interfaz estándar de acceso a capas inferiores de la torre de protocolos *Bluetooth*. El objetivo principal de este interfaz es la transparencia, de modo que el transporte (es decir, la conexión electrónica o interfaz entre el dispositivo *Host* y el *Controler*) sea independiente de las capas altas de la pila Bluetooth. Esto permite modificar las características de dicha interfaz sin que ésto afecte al resto de la comunicación. La norma prevé distintos protocolos de transporte para diferentes interfaces de conexión, como USB, SPI, UART, etc. En el caso de que los protocolos *Host* y *Controller* residan en el mismo dispositivo (dispositivos de pila completa), el interfaz HCI puede ser puramente *software*.

Por medio de este interfaz se permite el envío y la recepción de datos hacia o desde otro dispositivo Bluetooth, realizar conexiones y desconexiones a nivel de Banda Base con otro dispositivo, activar el modo de bajo consumo, encriptación, autenticación, etc.

2.2.7. Nivel L2CAP

Este protocolo proporciona servicios de datos tanto orientados a conexión como no orientados a conexión a los protocolos y capas superiores, con capacidad de multiplexación, segmentación y reensamblado y abstracción de grupos. El protocolo L2CAP permite a los protocolos de niveles superiores y a las aplicaciones mantener varias comunicaciones simultáneas a través del mismo o diferentes enlaces lógicos, transmitiendo y recibiendo datagramas de hasta 64 kB, con posibilidad de soportar control de flujo y retransmisiones selectivas para cada comunicación.

Las principales características del protocolo L2CAP son las siguientes:

- Multiplexación de protocolo/canal: Durante la inicialización del canal, se utiliza esta capacidad para enviar la petición de conexión al protocolo superior correspondiente. Durante la transmisión de datos, la multiplexación de canal lógico se utiliza para distinguir las comunicaciones de las diferentes entidades superiores, ya que puede haber varias en curso de forma simultánea.

- Segmentación y reensamblado: Permite controlar la longitud de la PDU a transmitir, proporcionando las siguientes ventajas:

  - Permite intercalar datos de diferentes aplicaciones para satisfacer los requerimientos de latencia.
  - La gestión de memoria es más simple cuando L2CAP controla el tamaño del paquete.
  - La corrección de errores mediante retransmisión es más eficiente.
  - La cantidad de información que se destruye cuando una PDU L2CAP está corrupta o se pierde es menor que si se perdieran los datos de la aplicación.
  - La aplicación se desentiende de la segmentación necesaria para enviar la información a través de las capas inferiores.

- Control de flujo independiente para cada canal.
- Control de errores y retransmisión: Algunas aplicaciones requieren una tasa de error mucho menor de lo que la banda base puede proporcionar, de modo que se incluye esta funcionalidad en L2CAP, aunque el uso de la misma es opcional.
- Calidad de Servicio: El establecimiento de la conexión L2CAP permite el intercambio de información relacionada con la calidad de servicio esperada entre dos dispositivos Bluetooth. Cada implementación L2CAP monitoriza los recursos utilizados y asegura que dichos requerimientos se cumplen.

El estandar distingue desde la versión 1.2 varias entidades dentro del nivel L2CAP, que se muestran en la figura 2.14:

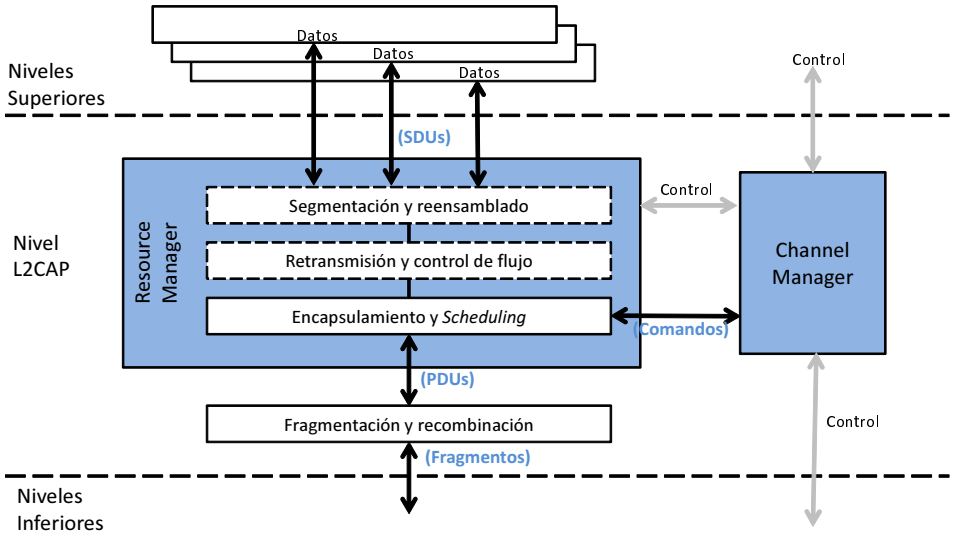


Figura 2.14: Diagrama de bloques de la capa L2CAP

- **Gestor de canales**. Se encarga de la creación, destrucción y gestión de canales L2CAP para el transporte de PDUs de protocolos de servicio o de datos aplicación, centralizando el intercambio de señalización interna tanto entre entidades homólogas L2CAP como con los niveles superiores. Si es necesario debe interactuar con el nivel *Link Manager* inferior para crear nuevos enlaces lógicos y configurarlos adecuadamente para obtener la calidad de servicio requerida.
- **Control de recursos**. Se encarga de gestionar la entrega ordenada de fragmentos de PDU al nivel banda base y de realizar la planificación de canales L2CAP para cumplir con los requisitos de QoS (*scheduling*). De forma optativa puede también realizar la fragmentación y reensamblado de las SDU de nivel superior en PDUs L2CAP de menor tamaño y ofrecer servicios de retransmisión y control de flujo por cada canal para aquellas aplicaciones que lo requieran.
- **Unidad de Fragmentación y recombinación**. Realiza la fragmentación y reensamblado de las PDUs L2CAP, contemplando la posibilidad de que la capacidad de

transmisión de los niveles inferiores sea limitada y requieran fragmentar a tamaños diferentes de las PDUs generadas en la segmentación L2CAP.

El nivel L2CAP soportaba desde la versión 1.2 hasta la 2.1 tres modos de funcionamiento diferentes, un modo básico equivalente al funcionamiento establecido en la versión 1.1 de la norma, en el que principalmente realiza la multiplexación de flujos, la gestión de la calidad de servicio y la segmentación; un modo con control de flujo, en el cual no se realizan retransmisiones pero sí se detectan paquetes perdidos; y un modo con retransmisiones basado en un mecanismo de ventana deslizante, en el que se utilizan temporizadores y PDUs de supervisión para garantizar que las PDUs de datos se entregan a la entidad homóloga. En el caso de activar el control de flujo, o las retransmisiones las PDUs utilizadas llevan un número de secuencia para identificarlas. Desde la versión 3.0 se dispone de dos modos adicionales, un modo mejorado con retransmisiones selectivas que utiliza un protocolo similar a HDLC, y un modo *Streaming* para tráfico isócrono que descarta paquetes cuando no se cumplen sus restricciones temporales.

Como ya se ha mencionado, según el modo de funcionamiento L2CAP puede realizar una doble fragmentación, como se muestra en la figura 2.15, ya que las SDU entregadas por el nivel superior se pueden dividir en PDUs de nivel L2CAP de menor tamaño, con sus correspondientes cabeceras, y éstas, antes de ser enviadas a los niveles inferiores, pueden volver a ser fragmentadas.

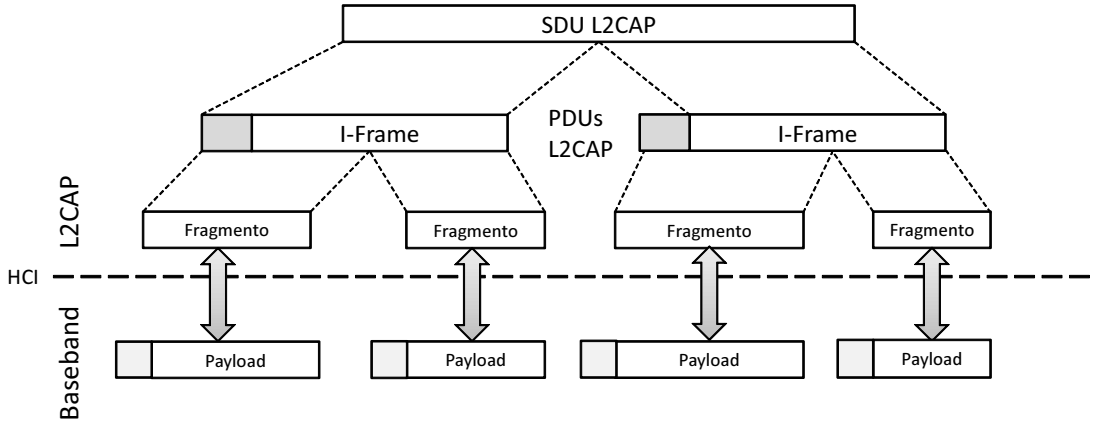


Figura 2.15: Fragmentación L2CAP en el modo con retransmisión

2.2.8. Calidad de servicio

Bluetooth define diferentes parámetros de calidad de servicio en las diferentes capas, si bien la especificación no define claramente cuáles son los mecanismos para garantizarla, de forma que cada fabricante tiene libertad en su implementación.

El gestor de recursos de la capa L2CAP es responsable de mantener la calidad de servicio entre los diferentes flujos de información, si bien los parámetros de calidad de servicio son negociados por los gestores de canales L2CAP. Los parámetros que se pueden negociar son:

- Tipo de servicio: *No traffic*, *Best Effort* y *Guaranteed*
- *Token rate*: representa la velocidad promedio de transferencia de datos
- *Peak bandwith*: representa la velocidad máxima de transferencia de datos
- *Token bucket size*: Representa la longitud máxima de una ráfaga de datos y por tanto el tamaño del buffer necesario para alojarla.
- *Latency*: Representa el valor máximo de retardo extremo a extremo requerido.
- *Delay Variation*: Máxima diferencia esperada entre los valores máximo y mínimo del retardo. Tiene un carácter meramente informativo.

Además hay un parámetro *Flush Timeout* orientado a tráfico isócrono que indica cuanto tiempo se va a esperar a que un paquete llegue exitosamente a destino. Los paquetes pueden ser descartados de forma automática por el nivel *Baseband* del controlador BR/EDR o bién pueden ser eliminados bajo petición del nivel L2CAP a través de los correspondientes comandos HCI, teniendo así un mayor control sobre lo que se descarta y lo que no.

En la especificación del L2CAP no se detalla cómo conseguir cumplir los requisitos de calidad de servicio, o implementar el control de admisión, y de hecho, se indica que únicamente es obligatorio soportar tráfico *best effort* sin garantías de calidad de servicio. A partir de la versión 3.0+HS se introdujeron algunas opciones extendidas de calidad de servicio, pero de nuevo su soporte sigue siendo optativo para los controladores BR/EDR y LE.

Si las entidades L2CAP son las responsables de mantener la calidad de servicio de los diferentes flujos de información, la capa *Baseband* es la responsable de mantener la calidad de servicio de las comunicaciones con los diferentes esclavos, y por tanto hay una dependencia entre ellas. A través del interfaz HCI la capa L2CAP puede indicar al *Controller* parámetros similares a los anteriormente mencionados (aunque no tienen por qué tener los mismos valores, ya que en su configuración hay que tener en cuenta la sobrecarga y tiempo de procesado introducidos por la capa L2CAP, la multiplexación de flujos sobre un mismo transporte lógico, etc.). Sin embargo, de nuevo el estándar no especifica cómo afectan estos parámetros al *scheduling* realizado por el nivel *Baseband*, y a la configuración de los parámetros de los diferentes enlaces. Excepto en el caso del *Flush Timeout*, que sí es un parámetro de funcionamiento del nivel *Basebadnd*, el resto de parámetros no lo son, ni se pueden negociar entre las entidades *Link Manager* del maestro y del esclavo. Además del ya mencionado *Flush Timeout*, el principal parámetro que determina la calidad de servicio del nivel *Link Manager* es la periodicidad con la que el maestro sondea a sus esclavos, T_{poll}. Este parámetro, que sí es negociado entre los *Link Manager* de maestro y esclavo, suele tener un valor por defecto de 40 *slots* (25 ms) en muchos dispositivos comerciales, y claramente afecta a la latencia de los paquetes que viajan desde el esclavo hacia el maestro. Aunque en el estándar no se indica expresamente, muchos dispositivos comerciales parecen fijar el valor de T_{poll} en función del parámetro de calidad de servicio *latency*.

2.2.9. Protocolo SDP y perfiles

El protocolo SDP proporciona los medios necesarios para descubrir los servicios disponibles en un dispositivo cercano y determinar sus características, en base a un modelo de petición y respuesta. Se trata de un protocolo simple que puede funcionar sobre un transporte no confiable, siempre y cuando el cliente implemente un mecanismo de *timeout* y repita las peticiones tantas veces como sea necesario.

El estándar *Bluetooth* define un cierto número de perfiles de aplicación (denominados perfiles *Bluetooth*) para definir qué tipos de servicios y aplicaciones ofrece un dispositivo. Además, con el fin de garantizar la interoperabilidad entre distintos dispositivos de diferentes fabricantes, la especificación define distintos casos de uso para cada perfil, detallando las características que son de implementación obligatoria y las que son opcionales, los protocolos en los que se basa y los procedimientos a seguir para proporcionar un servicio determinado.

2.2.9.1. Perfiles SPP y DUN. Protocolo RFCOMM

El perfil de puerto serie define los mecanismos que permiten proporcionar un servicio de emulación de comunicación por puerto serie RS232. El protocolo de transporte utilizado (por encima del nivel L2CAP) es RFCOMM, que está definido en el estándar ETSI TS 07.10. Este protocolo proporciona la emulación de las 9 señales definidas por RS232 y puede soportar hasta 30 conexiones simultáneas entre dos dispositivos. RFCOMM es un protocolo de transporte relativamente sencillo, ya que al apoyarse en el transporte fiable ofrecido por L2CAP, su funcionalidad radica en la multiplexación y control de los diferentes flujos de información (control y datos) que corresponden a las diferentes conexiones.

2.2.9.2. Perfil PAN. Protocolo BNEP

El perfil PAN define los medios para formar redes de datos entre dispositivos con interfaz *Bluetooth*, emulando el comportamiento de las redes de área local *Ethernet* y proporcionando soporte para protocolos de red comunes, como IPv4 e IPv6. El protocolo BNEP (*Bluetooth Network Encapsulation Protocol*) permite transmitir los paquetes Ethernet completos de un dispositivo a otro por el interfaz Bluetooth. Este perfil es el utilizado en la evaluación de las prestaciones de las *scatternets* presentada en la sección 2.5.

Dentro de este perfil se distinguen tres escenarios: la formación de una red punto a punto entre dos dispositivos que emula una comunicación *Ethernet* por cable cruzado, la creación de una red formada por un grupo de dispositivos en la que uno de ellos realiza el papel de *bridge Ethernet* para permitir la comunicación entre ellos, y por último, la implementación de un punto de acceso a red que permita a uno o varios dispositivos conectarse a una red externa de área más extensa a través de mecanismos de nivel 2 (*bridging Ethernet*) o nivel 3 (*routing* IP).

A su vez, se definen tres roles a nivel de aplicación para los dispositivos. Los dispositivos que ofrecen el servicio NAP (*Network Access Point*) y GN (*Group Networking*) son capaces de reenviar los paquetes *Ethernet* de los otros dispositivos y proporcionan por tanto la posibilidad de comunicarse entre varios de ellos, con la distinción de que un dispositivo

NAP dispone además de al menos un interfaz con una red externa a la cual da acceso. Los dispositivos que utilizan el servicio de los nodos GN o NAP para comunicarse entre ellos o acceder a una red, se denominan PANU (*PAN users*), y también pueden conectarse entre ellos para una comunicación punto a punto.

2.3. Trabajos relacionados

Bluetooth es una de las tecnologías inalámbricas más ampliamente difundidas y utilizadas para la transferencia de datos a corto alcance entre dispositivos. Su expansión ha acompañado al creciente desarrollo de los terminales móviles inteligentes y sus aplicaciones, y en la actualidad se encuentra desplegado en numerosos dispositivos, incluyendo tanto portátiles y tabletas como periféricos de diversa índole.

Debido a esta inmensa disponibilidad, *Bluetooth* es frecuentemente considerado una opción competitiva para la implementación de aplicaciones ubicuas que requieran la formación de redes expontáneas de área personal, llegando incluso a abarcar escenarios para los que esta tecnología no fue concebida originalmente y en los que a veces puede no obtener un rendimiento óptimo. Es por ello que la evaluación del rendimiento esta tecnología ha atraído un notable interés y ha sido objeto de diversos trabajos de investigación.

Como ya se ha mencionado, el esquema de funcionamiento básico de *Bluetooth* es una red con un pequeño número de dispositivos conectados (*piconet*) en el que uno de ellos asume el papel de maestro y coordina la actividad de los otros nodos, que actúan como esclavos. El maestro es el encargado de sondear cada cierto tiempo a sus dispositivos esclavos, que sólo pueden comunicarse con su maestro tras recibir de este el paquete de sondeo. La norma no especifica claramente qué algoritmos deben seguir los dispositivos maestros al planificar la comunicación con sus dispositivos esclavos para garantizar los parámetros de calidad de servicio que pueden ser especificados por las capas de nivel superior. Así pues, muchos estudios presentes en la literatura se centran en el estudio del comportamiento de las *piconets* y en la realización de diferentes propuestas para gestionar la división en el tiempo.

Para aumentar la conectividad a más de ocho dispositivos y permitir escenarios más complejos, *Bluetooth* introduce el concepto de *scatternet*, que se define como la unión de varias (dos o más) *piconets*. Esta unión se produce cuando un dispositivo que actúa como maestro o como esclavo en una *piconet*, es admitido como esclavo en una segunda *piconet*. Dicho nodo, que participa por tanto en la comunicación de ambas *piconets*, actuaría de puente entre ellas y se denomina *bridge*. Mediante la creación de varias de estas uniones a través de dispositivos *bridge*, la topología de la red podría ampliarse para expandir el área de cobertura y aumentar el número de dispositivos que participan en ella, lo cual puede ser conveniente en numerosas aplicaciones. La posibilidad de crear este tipo de redes plantea una serie de problemas adicionales, tales como la formación de la topología de red, el encaminamiento de la información por la red, y la planificación de las comunicaciones (*scheduling*) con los dispositivos *bridge*. Sin embargo, más allá de introducir y definir el concepto, la especificación de *Bluetooth* no proporciona directrices sobre cómo organizar, estructurar y gestionar las *scatternets*, por lo que también se han realizado multitud de

propuestas y estudios por parte de la comunidad investigadora sobre estos aspectos.

Como en otras muchas tecnologías inalámbricas, otro de los ámbitos de investigación importantes es el comportamiento y prestaciones de los mecanismos (FEC y ARQ) para asegurar la transferencia fiable de información en un medio en el que se pueden producir pérdidas por diferentes factores tales como la presencia de ruido e interferencias con otras comunicaciones, así como la interacción de dichos mecanismos con protocolos de nivel superior.

Finalmente, al ser una tecnología de baja potencia concebida para dotar de conectividad a dispositivos alimentados a batería, y que incluye diversos mecanismos y parámetros de configuración para reducir el consumo energético de los dispositivos, otra de las potenciales líneas de investigación se centra en la caracterización de dicho consumo en función de los diferentes parámetros de funcionamiento.

2.3.1. Estudios sobre las prestaciones en una *piconet*

Dado el esquema de control centralizado y planificado por división en el tiempo de *Bluetooth*, algunos autores han abordado el estudio del rendimiento de una *piconet* desde el punto de vista teórico de la ingeniería del teletráfico utilizando teoría de colas [MM03b] y/o modelado mediante cadenas de Markov [MZP04].

Entre ellos, [MM03b] modela la *piconet* como una cola M/G/1 y aborda el estudio del rendimiento de dos procedimientos de sondeo básicos, uno en el cual el maestro sondea a cada uno de los esclavos de forma consecutiva y permite el envío de un sólo paquete a cada uno (*limited round-robin*), y otro en el que el maestro permanece sondeando a un esclavo mientras éste indique que tiene paquetes que transmitir antes de pasar a sondear al siguiente (*exhaustive round-robin*). Estos mecanismos habían sido previamente propuestos y evaluados por simulación mediante eventos discretos para distintos tipos de tráfico en [CGK01]. El algoritmo de sondeo exhaustivo, aunque más eficiente, puede producir las conocidas situaciones de *starvation*, y en realidad incumple la especificación de *Bluetooth* al no garantizar un tiempo mínimo de sondeo, por lo que sus autores también proponen y analizan por simulación mecanismos de sondeo adaptativos. Propuestas similares son evaluadas de nuevo mediante análisis matemático en [MKM03], y los resultados obtenidos son aplicados por los mismos investigadores a diferentes cuestiones como el control de admisión [MCM04], la reserva de recursos para garantizar la calidad de servicio de tráfico CBR en presencia de tráfico asíncrono *Best Effort* [MMK04] o el estudio de las prestaciones de TCP [MM05] en base a modelos matemáticos del comportamiento del mismo [SKV03]. En la mayoría de estos trabajos el estudio analítico es contrastado y complementado mediante simulación por eventos discretos de modelos basados en redes de petri.

Más recientemente, otros autores han continuado esta línea de investigación. El análisis realizado por los trabajos anteriormente mencionados es ampliado en [ZSY07], donde se realiza un análisis con mayor profundidad del esquema TDD de *Bluetooth* que se aplica al estudio, de nuevo mediante teoría de colas, de diversos algoritmos de sondeo, tanto para tráfico simétrico como asimétrico. El modelo matemático se utiliza para obtener soluciones numéricas exactas para tráfico modelado como procesos de Poisson, que son validadas puntualmente por simulación basada en eventos discretos, y comparadas con los resultados

numéricos obtenidos aplicando el modelo de [MM03b]. Otros autores han trabajado proponiendo modificaciones sobre los algoritmos de planificación para mejorar su rendimiento en términos de *throughput* [HL10b] y consumo de energía [WN13], intentando mantener controlado el retardo extremo a extremo. En ambos casos se realiza en primer lugar un estudio teórico detallado del funcionamiento del mecanismo de sondeo propuesto, seguido de un estudio por simulación para evaluar las prestaciones del mismo. En el primer caso se utiliza un modelo del conjunto completo de la pila de protocolos para el simulador de eventos discretos NS-2, mientras que en el segundo se simula mediante MATLAB un modelo simplificado basado en cadenas ocultas de Markov derivado del análisis previamente realizado.

Las prestaciones de una *piconet Bluetooth* no sólo dependen de la política seguida al realizar el sondeo, y otros muchos autores se centran en el estudio del efecto de los mecanismos FEC y ARQ empleados para asegurar la entrega fiable de información. En lo que respecta a la protección FEC, ya se ha mencionado al describir la tecnología que el estándar *Bluetooth* define diferentes tipos de paquetes *Baseband* con distintos niveles de protección frente a errores y tamaño, que determinan su carga útil. En condiciones de bajo ruido obviamente resulta más conveniente utilizar los paquetes con mayor carga útil para maximizar el *throughput*, pero en presencia de mayor cantidad de ruido puede ser más conveniente utilizar paquetes con mayor nivel de codificación FEC para evitar que la mayor parte de los envíos sean infructuosos. Algunos autores como [CKSG04; CSC06; CTS02] se centran en el estudio del rendimiento de la codificación con objeto de proponer un algoritmo adaptativo que seleccione la codificación del paquete en función del tamaño de los datos a enviar y del nivel de ruido estimado en el canal. Otros muchos autores que siguen esta línea, presentan estudios que proponen el empleo de mecanismos de modulación y/o codificación que no están previstos en el estándar y que principalmente tendrían interés de cara a plantear futuras ampliaciones del mismo con una capa física y banda base alternativas.

Las prestaciones del mecanismo de retransmisión selectiva en presencia de ruido también ha sido estudiada por diversos autores. Por ejemplo, [Zan09a] plantea un estudio teórico en el que se asume un canal físico con desvanecimiento Rayleigh y se modela el comportamiento del mecanismo ARQ mediante un modelo de Markov de dos estados, del que mediante un análisis matemático se derivan algunas expresiones que permiten calcular el *throughput*, retardo y la eficiencia energética (desde el punto de vista radio exclusivamente).

Las pérdidas en *Bluetooth* no sólo pueden producirse debido al ruido introducido por la atenuación de la señal radio y sus desvanecimientos, sino que en muchas otras ocasiones la pérdida vendrá provocada por interacciones con otras tecnologías que operan en la misma banda, como IEEE 802.11, lo que dado la popularidad de ambas tecnologías también ha sido objeto de investigación por parte de diversos autores [Gol04; SSG07; LL09], mientras que otros estudios han abordado también las colisiones que pueden darse entre *piconets* vecinas [NWSS05; LL06; MHQ09]. Muchas de las propuestas para evitar las interferencias se basan en utilizar los mecanismos AFH (*Agile Frequency Hoping*) previstos por la especificación de *Bluetooth*, si bien otros muchos autores proponen métodos que son totalmente

ajenos a ésta y en algunos casos frontalmente incompatibles.

Además de estudios teóricos centrados en el estudio del comportamiento de un elemento concreto de la pila de protocolos, también pueden encontrarse en la literatura otros estudios que intentan realizar una evaluación más global de las prestaciones de una *piconet* bajo diferentes condiciones de funcionamiento, en este caso normalmente mediante el uso de simuladores o dispositivos reales. Así por ejemplo, [WW08a] realiza una evaluación empírica de la comunicación entre maestro y esclavo, proporcionando medidas del retardo de transmisión y de la tasa de error de paquete, [HAMMG09] evalúa las prestaciones de la transmisión de un fichero en función de la distancia entre los nodos y con presencia de obstáculos, [PPMF09] evalúa las prestaciones y consumo energético de la transmisión de datos entre terminales móviles utilizando la API de gestión de energía proporcionada por el software del teléfono, y [ADD09] comprueba la viabilidad de utilizar *Bluetooth* para la localización por difusión de mensajes. El inconveniente obvio de este tipo de estudios es que resulta complejo llegar a controlar la forma en la que se producen las pérdidas, las interferencias con otras tecnologías, y otros factores que pueden también afectar al rendimiento.

La alternativa seguida por otros estudios es evaluación mediante simulación basada en eventos discretos de modelos exhaustivos que incluyen la implementación de las diferentes capas de protocolos con un gran nivel de detalle. En la tesis doctoral [CB10] se estudia mediante un modelo de simulación para NS-2 el efecto del tiempo de poll (T_{Poll}) en las prestaciones de una *piconet* con presencia de tráfico multimedia, proponiendo posteriormente un algoritmo para ajustar dinámicamente dicho parámetro en función de las necesidades del tráfico en cada momento, y por último se evalúa una propuesta de algoritmo de *scheduling* que mejora las prestaciones de los algoritmos básicos mencionados anteriormente. Los resultados de estos estudios se recogieron más tarde en artículos de revista [CC11b; CC11a]. En dicha tesis también se incluye una interesante comparativa entre los resultados obtenidos mediante simulación y los obtenidos mediante un experimento real para el mismo escenario de red con una topología sencilla, y se constata la presencia de una disparidad significativa en algunos resultados, como por ejemplo el *jitter*. Los mismos autores han realizado muy recientemente un estudio experimental que analiza las prestaciones del procedimiento de descubrimiento de dispositivos (*Inquiry*) con objeto de seleccionar estrategias de configuración de sus parámetros que permitan mejorar la eficiencia del procedimiento cuando muchos dispositivos están intentando encontrar a otros o ser encontrados [CC15].

En la tesis doctoral [MF08] se defiende un interesante estudio que incluye un modelo derivado de un análisis detallado del funcionamiento de los diferentes protocolos, y un conjunto de pruebas empíricas muy exhaustivas para validarlo. El modelado se realiza manteniendo una total fidelidad a las especificaciones de *Bluetooth* e incluyendo observaciones sobre el comportamiento de los dispositivos obtenidas en experimentos reales, y de él se derivan unas fórmulas que permiten computar de forma numérica el retardo de los paquetes en sentido uplink y downlink en función de su tamaño y de la tasa de pérdidas. El modelo se valida mediante una red con topología sencilla (un maestro y un esclavo) sobre la que se realiza una batería de pruebas muy exhaustivas para comprobar que el

modelo ofrece unos resultados correctos para los diferentes parámetros. En este estudio las pérdidas son provocadas mediante una red *WiFi* interferente, y se estiman utilizando las estadísticas *Link Quality* reportadas por el dispositivo Bluetooth. Los resultados parciales de estos estudios se recogen en diversos artículos de revista [MLCDE07; MLCDE08; MLCDE09].

La tesis doctoral [LG10], continúa el estudio anterior y lo amplía a la tecnología EDR de *Bluetooth*, intentando mejorar además diversos aspectos metodológicos. Con el objetivo de realizar un mejor control de las pérdidas, éstas son producidas mediante un atenuador y un generador de ruido blanco gaussiano que se unen al dispositivo transmisor y receptor mediante un cable coaxial, lo que evita además que se producen interferencias no controladas debidas a otras tecnologías que operen en la misma banda. La fórmula derivada del estudio analítico considera una tasa de pérdida de paquetes (PER), por lo que se combinan estudios ya existentes realizados por otros autores sobre las diferentes modulaciones de *Bluetooth* con un estudio de las posibles codificaciones FEC para derivar la tasa de error para cada tipo de paquete en función de la tasa de error de bit, y ésta a su vez de la relación entre potencia de señal y de ruido. De nuevo, los resultados numéricos obtenidos mediante el análisis teórico es contrastado con pruebas exhaustivas realizadas en el entorno de pruebas con dispositivos comerciales. Los artículos [LMC10; MLC10; LMC12; LMC15] recogen diferentes resultados de estos estudios.

2.3.2. Estudios sobre *scatternets*

Las *scatternets* de *Bluetooth* atrajeron ya desde la aparición de las primeras versiones de las especificaciones el interés de la comunidad investigadora, espoleada por las posibilidades que podrían brindar en muchas aplicaciones y la escasez de directrices y orientaciones que ofrece el estándar sobre ellas, dando lugar a multitud de estudios sobre los diferentes aspectos de su gestión. En [PMS05; WHC05; SZ06] se recogen y clasifican muchas de las primeras propuestas, principalmente focalizadas sobre la formación de la topología de red y encaminamiento de la información.

En muchos de estos estudios, las prestaciones de la propuesta son evaluadas en términos abstractos, tomando como métricas que determinan la bondad de la *scatternet* formada parámetros como:

- Número de *piconets*.
- Número medio de esclavos por *piconet*.
- Número medio de roles por dispositivo.
- Número medio de *bridges* por *piconet*.
- Número de *bridges* S/M.
- Longitud media del camino más corto.
- Tiempo de formación de la red.

- Diámetro de la red.

En general, cuando se crean nuevos algoritmos de formación de *scatternet*, los distintos autores buscan cumplir uno o varios de los siguientes objetivos:

- Conectividad completa de la *scatternet*.
- Maximizar el ancho de banda teórico.
- Minimizar la longitud media de enrutamiento.
- Maximizar la disponibilidad media de los nodos.
- Minimizar la sobrecarga de conmutación.
- Autorreparación de la red ($Self - Healing$).
- Formación de *scatternet* bajo demanda.

Un criterio importante para clasificar los estudios existentes es la topología de red que se persigue al implementar el algoritmo de formación y el tipo de nodos *bridge* que se admite. En una *scatternet*, un dispositivo *bridge* puede ser maestro para una de las *piconet* y esclavo para la otra, lo que a partir de ahora denominaremos como *bridge* S/M. Otra posibilidad es que dispositivo *bridge* sea esclavo para diferentes maestros (*bridge* S/S). En ambos casos el *bridge* debe ser capaz de interactuar correctamente con el resto de equipos, por lo que en el caso S/M debe ser capaz de gestionar las comunicaciones con su maestro a la vez que de vez en cuando continúa sondeando a sus esclavos, y en el caso S/S debe ser capaz de aceptar comandos procedentes de diferentes maestros y mantener la sincronización con cada uno de ellos. También podría darse un escenario híbrido (nodo *bridge* S/M/S) en el cual el dispositivo es esclavo de más de un maestro y a su vez mantiene su propia *piconet* con uno o varios esclavos. Tal y como se menciona en las revisiones bibliográficas citadas anteriormente [PMS05; WHC05], algunos autores tienen en cuenta que mantener la sincronización con muchas *piconets* es inviable para un dispositivo con recursos limitados, por lo que diseñan su algoritmo restringiendo el *número de roles* que un dispotivo puede tener (es decir, el número de *piconets* a las que puede pertenecer con distintos roles), pero otros muchos no lo hacen.

En función del tipo de *bridge* que admiten o priorizan y de las limitaciones impuestas a éstos, así como de otras restricciones habituales como por ejemplo el número de puentes que puede existir entre dos *piconets*, pueden definirse una serie de topologías básicas: topología en arbol (*tree*) [ZBC01; MC04; PK04; MGS+05; Beu05b; LCXW11], topología en malla (*mesh*) [PBC03; PBC04; LSW04; ZSD08], y topología en anillo (*ring*) [FC02; LTCT03; SAKD12; HKC08]. Cada topología a su vez puede implementarse, según la propuesta, mediante *bridges S/S* o *bridges S/M*, como se muestra en la figura 2.16.

Además de los estudios centrados en la topología de red, otro campo importante de investigación es el problema de la planificación de las comunicaciones en los nodos *bridge*, que ha dado también lugar a un buen número de estudios. Por ejemplo, [YTD+03] propone una nueva política de *scheduling* para reducir el desperdicio de tiempo de enlace

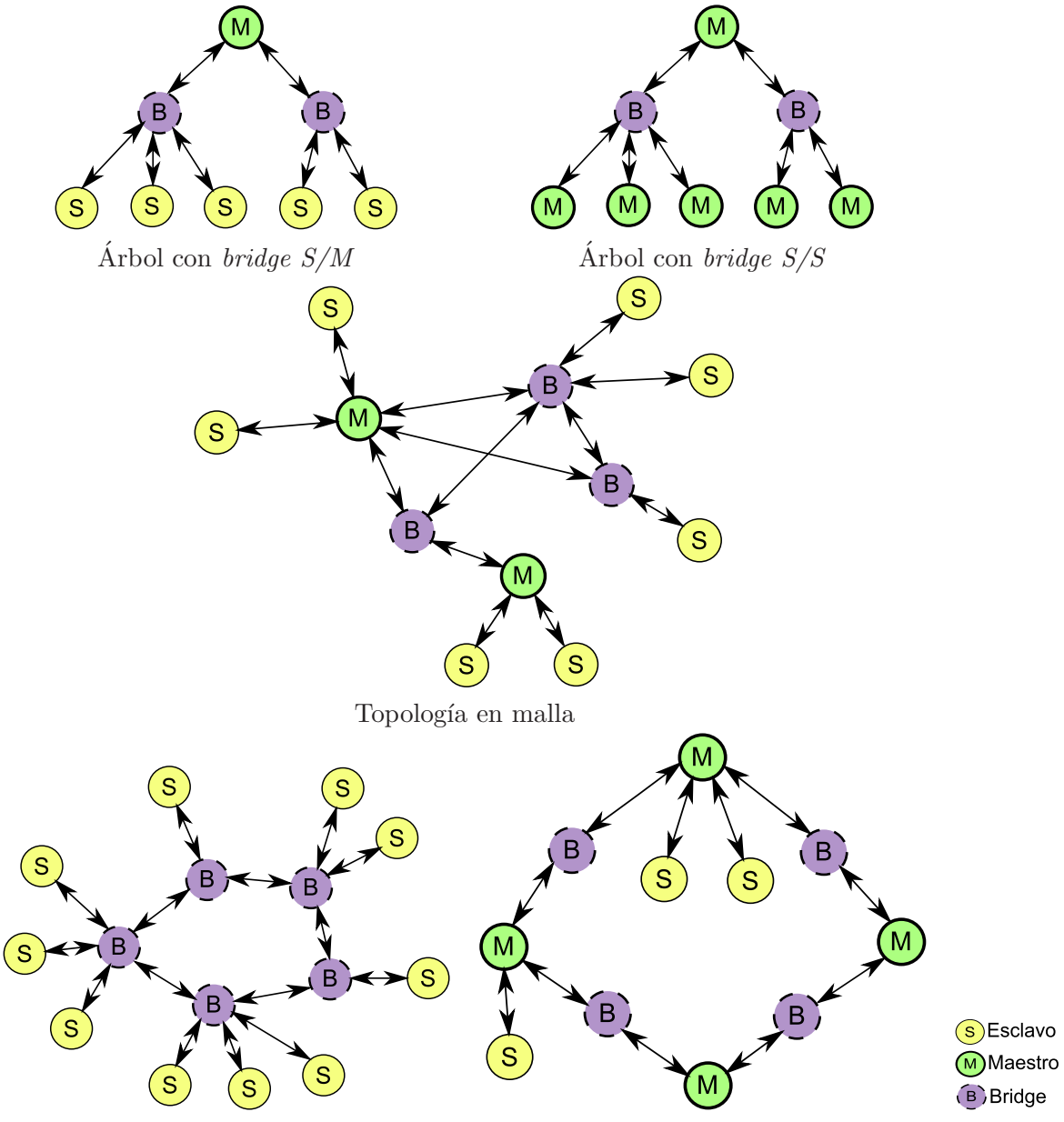


Figura 2.16: Topologías *scatternets* con diferentes tipos de *bridges*

producido por la conmutación entre *piconets* de los nodos *bridge*. [MM03a] aborda el estudio del comportamiento de los nodos *bridge* (tanto S/S como S/M) en las *scatternets*, siguiendo la misma metodología basada en análisis matemático mediante teoría de colas combinada con simulaciones que sus autores emplearon en los estudio de *piconets* previamente citados en la sección 2.3.1, centrándose de nuevo en las estrategias de planificación básicas (*round-robin* exhaustivo y limitado) ya estudiadas en el otro escenario. Siguiendo la misma metodología, los mismos autores también analizan las prestaciones de los mecanismos de sondeo adaptativo en los dos escenarios posibles, con y sin limitación en las colas de transmisión [MMR05; MM04b; MRM06]. Los mismos autores analizan en [MM04a] las ventajas de utilizar un algoritmo de planificación que contemple un posible preacuerdo entre los nodos (*rendezvous*) en un *bridge M/S* de una *scatternet* sencilla, aunque ellos mismos reconocen que la escalabilidad de esta opción en redes más compleja es poco viable. Los autores en [LLS03] sugieren alinear la temporización de todas las *piconets* a lo largo de cada ruta. [RB08] señala el hecho de que en una *scatternet* los maestros desperdician muchos *timeslots* al ser obligados a sondear a esclavos que permanecen inactivos o generan escaso tráfico, y para contrarrestarlo proponen un algoritmo de planificación que asigne *slots* a maestros y esclavos dependiendo del estado de las colas en los nodos, que se evalúa mediante simulación con NS-2. [RKSS07] presenta una propuesta de algoritmos de sondeo específicos para *scatternets* junto con un algoritmo para la construcción dinámica de su topología.

En la inmensa mayoría de los estudios sobre *scatternets* no se realiza ninguna prueba experimental del algoritmo propuesto que aporte evidencias de que puede funcionar con dispositivos reales, y las conclusiones están normalmente basadas en estudios analíticos o en simulaciones en las que no se emula el funcionamiento de los diferentes protocolos de *Bluetooth* o se modela de forma muy simplificada, con algunas excepciones que sí inciden en la importancia de utilizar un modelo más detallado posible [JCJM14]. Muchas de los planteamientos asumen o proponen utilizar funcionalidades que requerirían como poco la ampliación del estándar con nuevos tipos de paquetes y esquemas de acceso al medio. Por ejemplo, [SBTL05; CAA05] toman ventaja de la posibilidad de establecer una comunicación directa entre dos nodos cualquiera de una misma *piconet*. También se puede mencionar [HKC08] en el que la propuesta para la formación de la *scatternet* (denominada *Ring of Masters*) se basa en establecer un anillo de comunicación entre dispositivos maestros sin que estos sean esclavos unos de otros.

Por otra parte, en bastantes otras propuestas que aseguran específicamente ser compatibles con las especificaciones pueden encontrarse algunas inexactitudes debido a una interpretación no detallada de la normativa. Por ejemplo en [CC06] se propone la utilización adaptativa del proceso de *Role Switch* para mejorar la estructura de la *scatternet* dinámicamente mientras se construye, sin tener en cuenta que el procedimiento especificado en el estándar sólo permite a un dispositivo abandonar el papel de maestro si no tiene más hijos a su cargo, es decir, el *Role Switch* en realidad no permite la fusión de dos *piconets* si ambas tienen dispositivos esclavos. Si esto se intenta realizar en un experimento con módulos comerciales, el resultado es que el procedimiento no se realiza y se devuelve un error. Esto es lógico porque la fusión de dos *piconets* requeriría una reasignación de

las direcciones cortas y la sincronización de los dispositivos esclavos de una *piconet* con el maestro de la otra *piconet*, con quien no tienen comunicación directa. Igualmente, cuando el maestro de una *piconet* inicia un procedimiento de cambio de rol con uno de sus dispositivos esclavos, éste no queda como maestro de la *piconet* entera, sino que lo que se forma es una *scatternet* en la que el antiguo maestro tiene el papel de *bridge S/M* y el antiguo esclavo pasa a ser su maestro. La misma suposición de que se pueden fusionar dos *piconets* interconectadas en las que ambos maestros tienen esclavos, se utiliza también en [KCJ+07], donde se propone y evalúa un algoritmo para cambiar dinámicamente la topología de la *scatternet* con objeto de mantenerla y optimizarla.

Además de todo lo anterior, existen una serie de procesos que no están descritos en los estudios pero que son necesarios para la implementación real de las diferentes topologías, como puede ser la elección de un nodo inicial, que no se describe en casi ninguno de los estudios enumerados. Si bien la elección del nodo que comienza a realizar la formación de la red es factible en un entorno de simulación, en un entorno real distribuido quedaría por solucionar la forma en la que todos los nodos se ponen de acuerdo para que sólo uno de ellos comience a formar la red. Otra de las asunciones más comunes es que cada nodo tiene pleno conocimiento del resto de nodos que intervienen en la comunicación, y otra es que todos están en rango.

Sólo algunos estudios presentes en la literatura realizan pruebas en *testbed* reales para validar su propuesta o asegurar su viabilidad en aplicaciones comerciales, y por lo general los resultados difieren significativamente de lo predicho por los estudios analíticos y las simulaciones. En [PPV07] los autores investigan las prestaciones de una política concreta de formación de *scatternets* (denominada *Bluepleiades* [DHM+07]) mediante la realización de una serie de ensayos experimentales utilizando dos plataformas *hardware* diferentes. Sus resultados revelan que la evolución de la formación de la *scatternet* es peor que la predicha tanto por el estudio teórico como por las simulaciones. Los autores achacan este hecho a la falta de soporte de algunas características opcionales del estándar por parte de los módulos comerciales. Por otra parte, en [KA05] se describe un algoritmo de formación orientado a una topología tipo árbol, y se valora su funcionamiento en una red experimental. Puesto que el *hardware* comercial utilizado no soporta algunas de las caracterísicas requeridas, en las pruebas reales se tienen que emular utilizando el modo *hold* y se obtienen unos resultados bastante más pobres que por simulación.

Los autores del interesante estudio recogido en [JCG06] desarrollan una herramienta denominada *BlueProbe* para estimar el *throughput* en un camino a través de una *scatternet* real, con varios saltos, que persigue contrastar los resultados del estudio analítico realizado por [KBG05]. Se prueban varias topologías de red y el efecto de tráfico cruzado, y se concluye que los nodos S/S pueden degradar las prestaciones más de lo previsto en los estudios analíticos mencionados, llegando incluso a sugerir que para ciertos tipos de tráfico resulta más eficiente el establecimiento de conexiones temporales entre los nodos que el mantenimiento de una *scatternet*. Esta observación es coherente con los resultados que mostramos más adelante en este capítulo y contrasta con que muchos protocolos tradicionales [PBC03; PBC04; LSW04; DHM+07] de formación de red favorezcan la utilización de puentes S/S como elemento de unión entre las *piconets*.

En los trabajos [Beu05b; BDMT05], recogidos con mayor nivel de detalle en la tesis doctoral [Beu05a], se lleva a cabo el diseño *hardware* de un nodo para redes de sensores con interfaz *Bluetooth* y entorno TinyOS, que se utiliza para realizar la demostración en vivo de una red de sensores con tecnología *Bluetooth*, llegando a desplegar una *scatternet* de más de 50 nodos. En dicha red se realizan algunas medidas de tiempo de ida y vuelta *Round Trip Time* en función del número de saltos, obteniendo que la media es de 42 y 60 ms por salto para paquetes de 10 y 117 bytes respectivamente. A este respecto hay que señalar que el reenvío de la información de un nodo a otro en una plataforma con baja capacidad de proceso (microcontrolador de 8 bits) y conectada al módulo *Bluetooth* por un interfaz serie puede introducir cierto retardo debido a la propia plataforma y no achacable directamente a Bluetooth. El algoritmo de formación utilizado en estos trabajos genera una topología de tipo árbol con *bridges S/M*.

En [EMA14; EA15] se desarrolla una metodología para la medida del retardo de transmisión en cada salto de un paquete en su viaje a través de una ruta que atraviesa distintos nodos de una red de pruebas real, y se aplica para obtener un modelo empírico del retardo que permita estimar éste en función del número de saltos en una red de varios niveles con *bridges S/M*. El retardo en cada salto se mide con una buena precisión y sin que interfieran retrasos adicionales debido al reenvío de la información en cada nodo, pero el problema de la metodología empleada es que es difícilmente escalable a tráfico más complejo que el estudio de un único paquete viajando por la red cada vez (es decir, no se inyecta un nuevo paquete hasta que el anterior haya salido de la red), ya que se emplea un analizador lógico para medir el tiempo entre pulsos de una señal de disparo que generan los nodos conforme el paquete los va atravesando.

2.3.3. Estudios sobre el comportamiento energético

Puesto que la tecnología *Bluetooth* se utiliza frecuentemente en dispositivos periféricos alimentados a batería, y su uso ha sido incluso propuesto para aplicaciones de redes de sensores, otro de los temas de interés se centra en trabajos para la estimación y optimización del consumo energético. Por consumo debemos entender la energía gastada en alimentar el dispositivo de comunicaciones *Bluetooth*, que no debe confundirse con la energía emitida por la radio, sino la energía requerida por el transceptor para su operación. En un dispositivo de baja potencia de transmisión como son los transceptores *Bluetooth*, resulta frecuente que la energía necesaria para mantener al dispositivo en modo recepción (es decir, con los amplificadores RF, osciladores, mezcladores, demoduladores, y demás circuitería necesaria para realizar la recepción, adecuadamente alimentados) sea sólo ligeramente inferior a la necesaria para realizar la transmisión. Esto hace que a pesar de lo que pudiera pensarse en un principio, en *Bluetooth* los dispositivos maestros tengan habitualmente un perfil de consumo más reducido que los esclavos, ya que tienen el control de la comunicación y pueden apagar la radio en los *slots* que no van a utilizar, mientras que los esclavos, a menos que entren en uno de los posibles modos de bajo consumo, permanecen todo el tiempo con la recepción habilitada. La caracterización del estudio del consumo energético es importante de cara a establecer el tiempo útil que puede permanecer funcionando un dispositivo alimentado a baterías antes de tener que proceder a la sustitución o recarga de

las mismas.

La mayoría de estudios de consumo se basan en el análisis y caracterización de algún dispositivo comercial, que si bien pueden presentar diferentes valores cuantitativos de consumo en los diferentes estados, suelen presentar un comportamiento cualitativo bastante similar. Existen distintas estrategias a la hora de estimar y/o modelar el consumo:

- Utilización de un multímetro digital de integración contínua como el Agilent 34401/34410 [MP07; MN06]. El método más preciso de estimación del consumo, aunque requiere un instrumento especializado dedicado que supone una cierta inversión.

- Utilización de un circuito amplificador de transimpedancia y un sistema de adquisición de datos de bajo coste. Es el procedimiento que se utiliza en las medidas de consumo presentadas en este capítulo y que posteriormente ha sido utilizado por otros autores, utilizando incluso el mismo amplificador INA193 propuesto por nosotros en [CCG+06a; CCG+06b] para realizar medidas similares en módulos *Bluetooth* más recientes [BECL08; EBL+12]. La desventaja es que la precisión de la medida es inferior a la del multímetro, y normalmente se requiere un calibrado del sistema para evitar errores de *offset*, ganancia y no linealidad. Por contra, el coste es mucho menor y resulta más factible realizar la monitorización de un mayor número de dispositivos.

- Utilización de una resistencia de *shunt* y de un osciloscopio para realizar medidas de la forma de onda de la corriente [NT06; NBD06; FKK13]. Al tratarse de un instrumento de medida, su coste de nuevo es mayor (aunque cada vez resulta más habitual encontrar en el mercado instrumentación de coste bastante asequible). Normalmente no permiten hacer medidas contínuas en tiempo real, sino que realizan capturas de ventanas temporales. La precisión también es inferior a la obtenida utilizando un multímetro.

- Utilización de un perfilador (*profiler*) de energía en un *smartphone* o dispositivo similar [PAGW06]. Este tipo de herramienta permite tomar *medidas* del consumo a través de la API del dispositivo (normalmente son estimaciones en función del tiempo dedicado a cada actividad y el gasto de batería observado).

Algunos estudios que presentan resultados de consumo en función de algunos parámetros no indican cómo ni bajo qué condiciones han realizado la medida [YZJ07].

En cuanto a los aspectos estudiados en los diferentes trabajos mencionados, los artículos[MP07; MN06] se centran en el desarrollo de un sistema de instrumentación automatizado para medir el consumo en una conexión *Bluetooth* punto a punto, realizando un análisis muy exhaustivo de las posibles fuentes de error en la medida con objeto de optimizar el tiempo necesario para estimar adecuadamente el consumo promedio en cada estado modelado. Los módulos estudiados son los Eriksson ROK 101008, que son los primeros módulos *Bluetooth* que estuvieron disponibles comercialmente y tenían muchas limitaciones (entre otras cosas, sólo admitían conexiones punto a punto), además de presentar un consumo muy elevado incluso en modo *standby*.

En [NT06; NBD06; CCG+06a; CCG+06b] se analizan los comportamientos en distintas situaciones del dispositivo maestro y los dispositivos esclavos, considerando también los modos de bajo consumo y en particular el modo *sniff*. Los resultados cuantitativos obtenidos dependen del *hardware* concreto, pero en general parecen indicar que el consumo del maestro es inferior al del dispositivo esclavo en una conexión punto a punto, y se va incrementando de forma lineal al incrementar el número de dispositivos esclavos que forman parte de la *piconet*. De la misma forma, el consumo en modo *sniff* es lineal con respecto al ciclo de trabajo.

En [BECL08] se analizan de nuevo estos mismos escenarios y algunos adicionales, como por ejemplo la energía empleada en el procedimiento para el establecimiento de un determinado número de conexiones, el consumo de maestro y esclavo variando parámetros de *QoS* como el tiempo T_{poll}, o el consumo en función de la intensidad del tráfico. En [EBL+12] se propone y valida empíricamente un modelo bastante complejo para estimar el consumo de dispositivos en modo *sniff* con un bajo ciclo de trabajo en función de su *rol*, del número de conexiones, de los parámetros de *sniff* y del tráfico de datos (tamaño y número) de los paquetes que cursan. Los parámetros del modelo, alguno de los cuales no tiene un significado físico muy claro, se ajustan experimentalmente.

En [FKK13] se analiza el consumo de diversos modelos de teléfono inteligente realizando diferentes operaciones con los interfaces *Bluetooth* y *WiFi*, con objeto de determinar cuál resulta más eficiente en diferentes operaciones (envío de un fichero, *streaming* de audio, etc.). Al tratarse de medidas realizadas sobre un sistema completo, no es posible aislar el consumo del interfaz *Bluetooth* del resto de elementos sistema, aunque se puede realizar una estimación restando el consumo base medido para el mismo con el interfaz desactivado.

Las medidas de consumo, y los modelos de comportamiento, derivados de estudios experimentales son a menudo utilizadas en simulaciones para realizar estimaciones del coste energético como métrica de bondad para evaluar la alternativas propuestas (por ejemplo en estudios como [WN13; Tan11; MPS+09]) o como criterio para conmutar entre *Bluetooth* y otros interfaces de comunicación [PAGW06; FKK13].

Como ya se ha mencionado, no debe confundirse el consumo energético del módulo debido a su actividad, con la energía emitida por el transmisor o eficiencia energética espectral, sobre la que también existen estudios analíticos realizados [ZZ08; Zan09b] mediante técnicas de teoría de comunicación, y que viene determinada principalmente por las modulaciones y esquema de acceso al medio. Este concepto hace referencia a la eficiencia con que *Bluetooth* hace uso de la banda de frecuencias compartida para transmitir información.

2.4. Aportaciones al estudio y modelado del consumo de dispositivos *Bluetooth*

En esta sección se incluyen las aportaciones realizadas dentro del marco de prestaciones de consumo en redes *Bluetooth*, y que se han traducido en diversos artículos publicados en congresos y revistas científicas[CCG+06b; CCG+06a; CCG+07; CCM+08; CCGC+09].

2.4.1. Sistema de medida

La medida de la corriente se ha realizado mediante el montaje esquematizado en la figura 2.17, en el que el circuito integrado INA 193 es utilizado como amplificador de transconductancia para convertir la corriente a tensión, que posteriormente se muestrea y digitaliza mediante un sistema de adquisición de datos implementado mediante un microcontrolador de bajo coste conectado al PC que permite una frecuencia de muestreo de hasta 5 kilomuestras por segundo. El montaje basado en el INA193 incorpora también un filtro paso bajo que se ha configurado con una frecuencia de corte de unos 250Hz para evitar problemas de *alising* al muestrear la señal, sin afectar al valor medio estimado para la corriente.

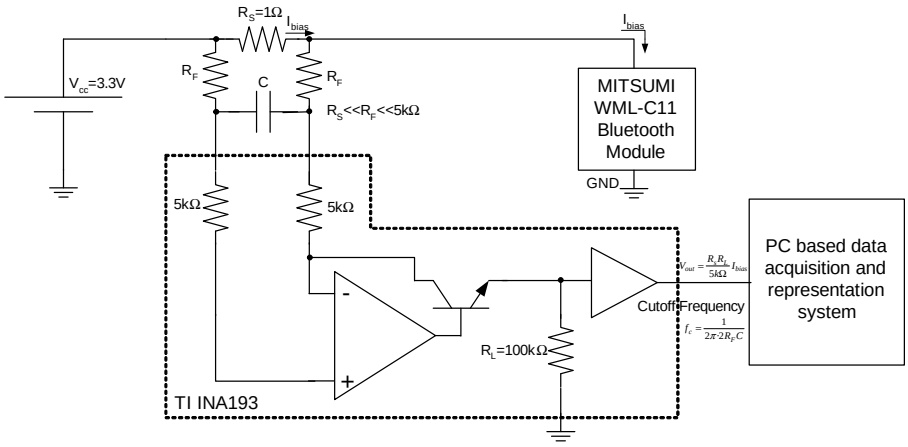


Figura 2.17: Circuito para la caracterización de corriente

Para corrientes bajas el INA193 muestra un comportamiento no lineal, por lo que antes de realizar las medidas de corriente el sistema debe ser calibrado. El proceso de calibración se realiza mediante el software de control desarrollado para controlar el sistema de adquisición, y consiste en medir diferentes niveles de corriente constante y compararlos con las medidas de un multímetro digital de seis dígitos y medio. Los intervalos establecidos se utilizan posteriormente en la medida para traducir a valores de corriente los valores de las muestras obtenidas. El *software* de control se encarga también de almacenar las muestras de corriente adquiridas en tiempo real.

El sistema de medida desarrollado muestrea por tanto en tiempo real la forma de onda de la corriente consumida por el dispositivo bajo prueba, y almacena las muestras adquiridas. La corriente media en un intervalo de tiempo T puede estimarse mediante el promediado de las N muestras adquiridas durante dicho intervalo (ecuación 2.1). En este sentido hay que tener en cuenta que el sistema cuenta con un filtro paso bajo que evita el *aliasing* y las variaciones rápidas de la señal sin afectar al valor medio, y por ello éste puede ser aproximadamente obtenido promediando las muestras.

$$\overline{I} = \frac{\int_T i(t)dt}{T} \approx \frac{\sum_N i[n]}{N} \tag{2.1}$$

2.4.2. Dispositivos evaluados

A lo largo de los estudios publicados en [CCG+06b; CCG+06a; CCG+07; CCGC+09], se utilizaron diferentes tipos de dispositivos *Bluetooth*. Como ya se ha mencionado, diferentes dispositivos comerciales suelen mostrar distintos valores absolutos de consumo en los diversos modos, pero un comportamiento cualitativo similar (las mismas formas de onda, que permiten definir los mismos estados). En general, gracias a la mejora de la tecnología electrónica de los dispositivos, aquellos más recientes suelen presentar un consumo menor al realizar operaciones de transmisión y recepción que los dispositivos más antiguos. Por otra parte, la tecnología que se utiliza para interconectar el módulo *Bluetooth* con el equipo que lo controla (PC o microcontrolador) tiene también un impacto importante en el consumo del módulo, siendo significativamente superior cuando la conexión es USB que cuando se realiza mediante un interfaz serie UART. Este consumo, que está presente incluso cuando el módulo está en el estado *stand-by*, es independiente del consumo debido a la propia actividad de *Bluetooth*, y por tanto en las medidas puede considerarse como un *offset* o desplazamiento añadido en los módulos que utilizan interfaz USB.

El primer dispositivo evaluado fue un prototipo de sensor inteligente con tecnología *Bluetooth* que se desarrolló en el marco del proyecto RIAM [Ria] por el grupo de investigación DIANA de la Universidad de Málaga. El diagrama de bloques se muestra en la figura 2.18, donde puede observarse que el sistema consta básicamente de un microcontrolador de bajo consumo conectado a un transceptor *Bluetooth* a través de un interfaz serie asíncrono (UART). El transceptor *Bluetooth* implementa las capas más bajas de la norma *Bluetooth*, es decir, las capas física, banda base (*Baseband*) y gestion de enlace (*Link Manager*), siendo el firmware del microcontrolador el que implementa las capas más altas. El microcontrolador configura y gestiona las operaciones del transceptor *Bluetooth* haciendo uso del interfaz HCI sobre puerto serie (mediante el protocolo H4) previsto por la norma. El transceptor *Bluetooth* es alimentado a través de un regulador de tensión que puede ser apagado por el microcontrolador para reducir el consumo del dispositivo sensor. Además se incluyó un subsistema para la monitorización del consumo de corriente del módulo que facilitase su estudio y optimización. En la figura 2.19 se muestra una foto del prototipo final.

Para optimizar el consumo y la duración de la batería se seleccionó un microcontrolador de la familia MSP430 de Texas Instruments, especialmente indicado para aplicaciones que requieren un bajo consumo. Por este motivo, esta familia de microcontroladores ha sido ampliamente utilizada en diferentes implementaciones de dispositivos inteligentes inalámbricos de bajo consumo con amplia difusión en la comunidad investigadora como el *TelosB* y el *TMote Sky* [Cro09]. Dentro del abanico de opciones ofrecidos por el fabricante, se seleccionó el microcontrolador MSPF430F1612 que cuenta con 55kB de memoria Flash y 5kB de RAM integrada. Como módulo *Bluetooth* se seleccionó el módulo de Clase 1 WML C11 de Mitsumi. Este módulo está basado en el chipset BlueCore2 de CSR y cumple con la especificación *Bluetooth* 1.2, si bien el firmware se puede actualizar para hacerlo compatible con la especificación 2.0 de *Bluetooth* sin soporte EDR (Enhanced Data Rate). Como se ha mencionado, el módulo *Bluetooth* se conecta al microcontrolador, y es

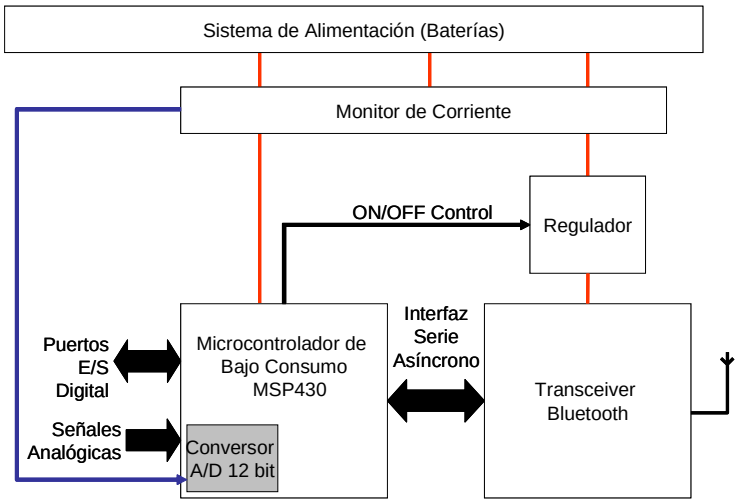


Figura 2.18: Diagrama de bloques del nodo inteligente de bajo consumo con interfaz *Bluetooth* desarrollado en el proyecto RIAM

controlado desde él, mediante el interfaz HCI sobre puerto serie. El microcontrolador a su vez puede conectarse por otro interfaz serie a un PC para controlar su operación.

Además de este dispositivo, otros módulos *Bluetooth* USB, basados en los chipset *Bluecore2* y *Bluecore4* de CSR y en un chipset de Broadcomm, han sido utilizados en algunas de las pruebas correspondientes a los resultados que se presentan en la siguiente sección. En este caso los dispositivos han sido conectados directamente a un PC y controlados mediante las herramientas y la API de programación que ofrece el sistema operativo Linux (*BlueZ*). A la hora de caracterizar un módulo comercial es importante tener cuidado con la presencia de elementos ajenos al funcionamiento de *Bluetooth* que pueden afectar al consumo. Un ejemplo típico podrían ser los *LEDs* indicadores que muchos módulos (especialmente *dongles* USB) incorporan y que parpadean siguiendo uno u otro patrón en función de la actividad del módulo. El consumo de dichos LEDs puede estar comprendido entre 1 y 5 mA, por lo que deben ser eliminados para evitar posibles desviaciones en la medida debido a su activación.

2.4.3. Pruebas y resultados

En esta sección se incluyen los resultados de las pruebas de consumo realizadas utilizando el sistema de medida y los dispositivos bajo estudio descritos en las secciones anteriores.

2.4.3.1. Inicialización y establecimiento de la conexión

En primer lugar se analiza el consumo del dispositivo durante su inicio tras ser habilitada su alimentación. En el caso del módulo *Bluetooth* Mitsumi WML C11 analizado, éste procedimiento dura unos dos segundos y finaliza cuando el módulo entra en estado *standby*, en el cual permanece mientras no se establezca una conexión *Bluetooth* o realice las actividades *inqury*, *scan*, *inquiry scan* o *page scan*. El consumo de corriente durante la inicialización se muestra en la figura 2.20, donde la línea azul representa el consumo

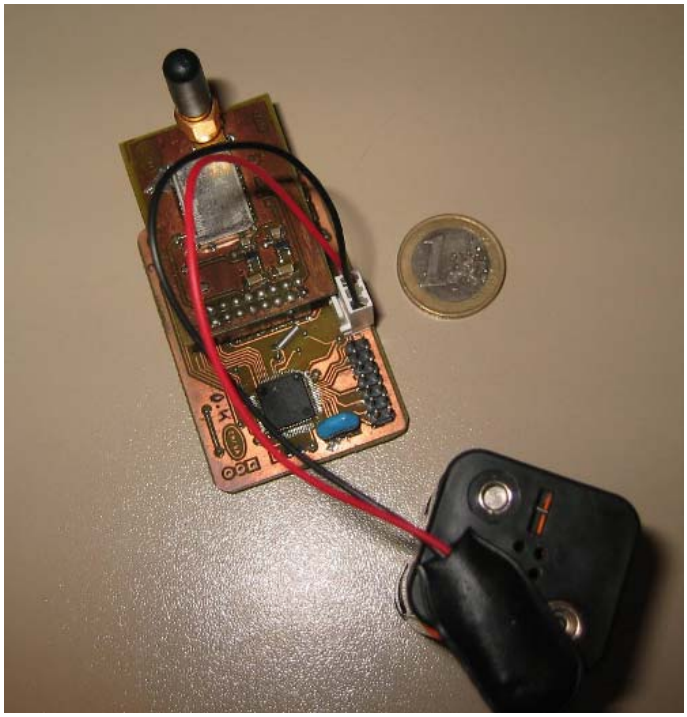


Figura 2.19: Prototipo de nodo inteligente de bajo consumo con interfaz *Bluetooth* desarrollado en el proyecto RIAM

instantáneo y la línea constante roja el consumo medio estimado durante el proceso, que es de unos 20mA. Esta estimación es de especial interés para aquellas aplicaciones en las que se considere apagar el módulo de forma periódica para ahorrar energía cuando no haya nada que transmitir.

El consumo en modo *standby* se ha observar en la figura 2.21, donde se ha representado tanto su valor instantáneo (línea azul) como el valor medio medido a lo largo de un intervalo de 5 minutos (línea roja), que es de aproximadamente 2mA. Este estado representa la cota inferior del consumo, y en el caso de dispositivos *Bluetooth* con interfaz USB puede llegar a arrojar valores elevados de entre 6 y 9 mA.

Como se detalla en la sección 2.2.3.9, cuando un dispositivo *Bluetooth* se configura como *conectable* y/o como *descubrible*, periódicamente abandona el estado *Standby* y entra temporalmente en los estados de *Page Scan* o *Inqury Scan*, en los que permanece escuchando en una frecuencia concreta. La duración de estos estados y su periodicidad es configurable, e incluso se puede programar el módulo para que permanezca continuamente en dichos estados, de forma que se pueda caracterizar mejor el consumo medio de corriente en ellos. En ambos casos el comportamiento es bastante parecido. En la figura 2.22 se muestra el consumo del módulo en estado de *Page Scan*, en el que consume unos 45mA de media.

Los procedimientos de *Inquiry* y *Page* son los utilizados para realizar la búsqueda de dispositivos vecinos, y conectar con ellos. La mecánica de ambos procedimientos es similar, aunque el procedimiento de *inquiry*, que se muestra en la figura 2.23, tiene una duración más extendida.

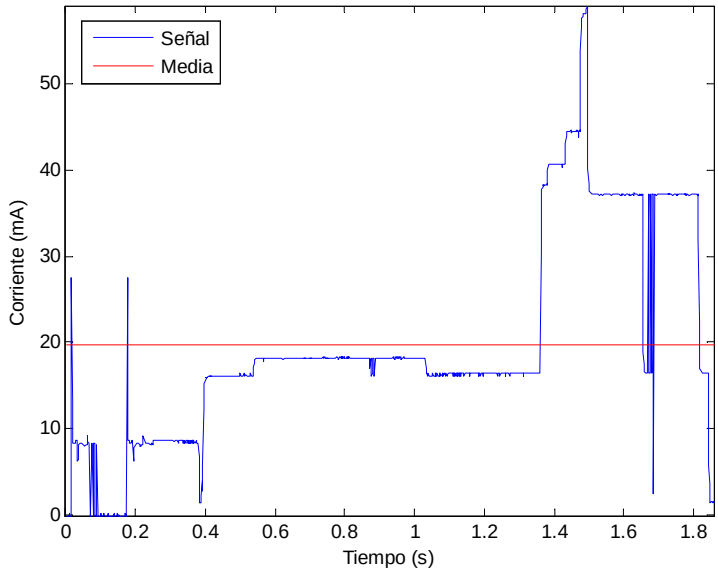


Figura 2.20: Consumo durante el encendido del módulo *Bluetooth*

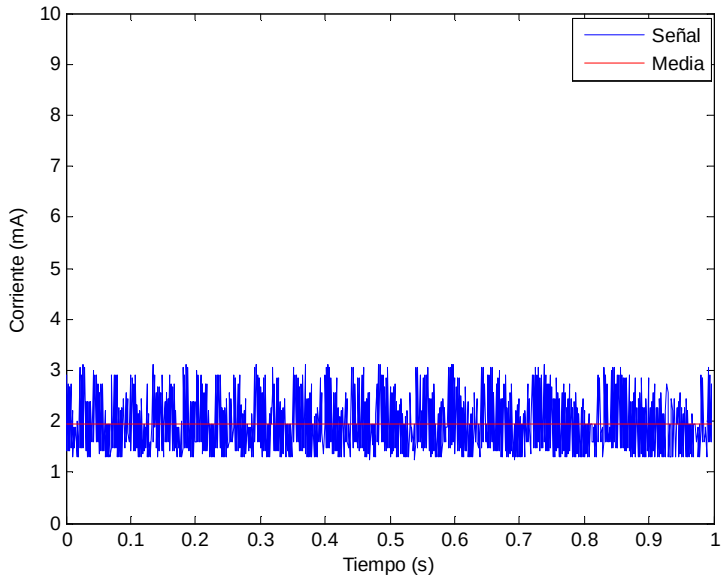


Figura 2.21: Consumo del dispositivo en modo *Standby*

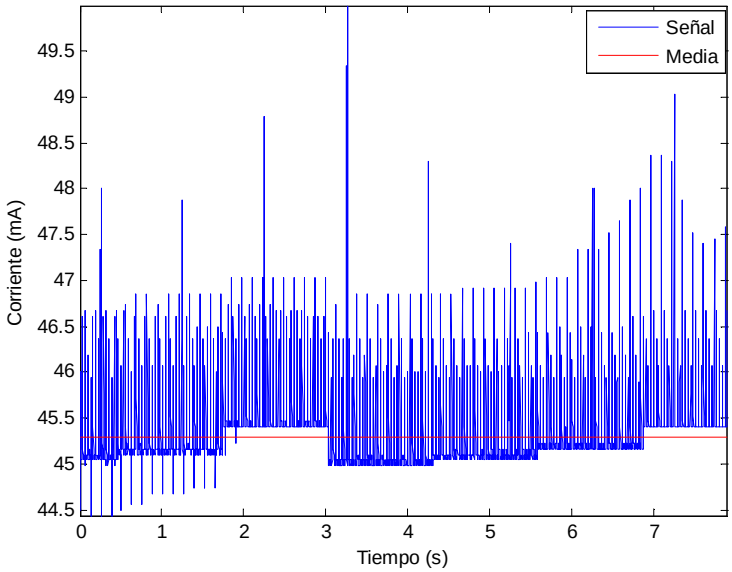


Figura 2.22: Consumo durante el estado *page scan* (extendido)

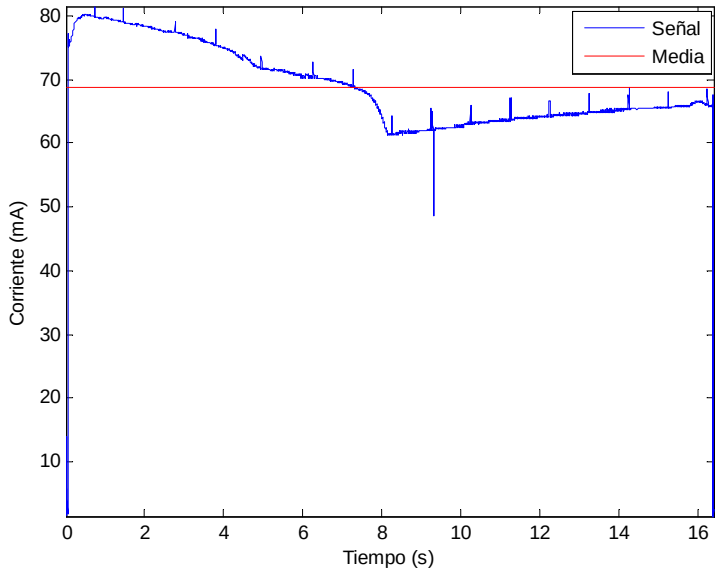


Figura 2.23: Consumo del dispositivo al realizar un *Inquiry*

2.4.3.2. Consumo de maestro y esclavo

Una vez que queda establecida la conexión, un dispositivo puede tener el papel de esclavo o el de maestro de una *piconet*. En las figuras 2.24 y 2.25, se muestran la corriente instantánea medida para un dispositivo participando en una *piconet* con los papeles de esclavo y maestro respectivamente, así como el valor de corriente promediado a lo largo de 10 minutos de conexión. Debido a que el maestro controla la temporización del enlace, puede apagar la radio (desactivar el receptor) cuando no va a sondear al dispositivo esclavo, y por tanto su consumo medio (20mA aproximadamente) es bastante inferior al del esclavo (41mA).

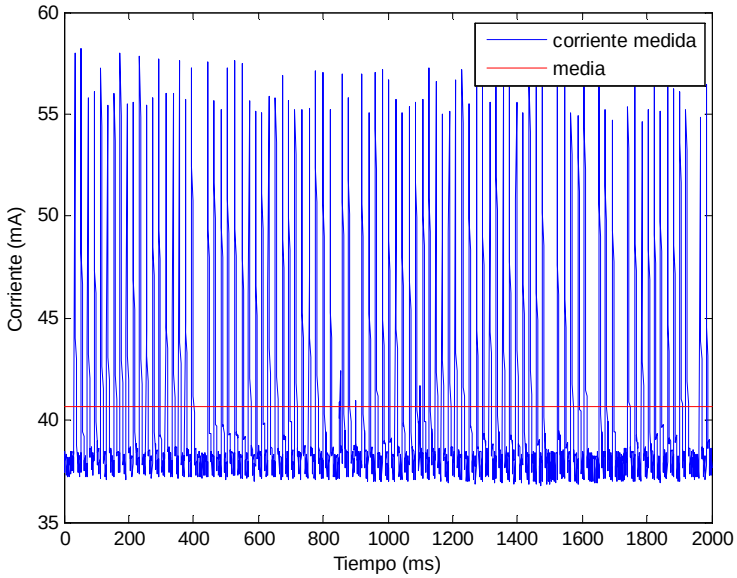


Figura 2.24: Consumo del dispositivo participando en una *piconet* como esclavo

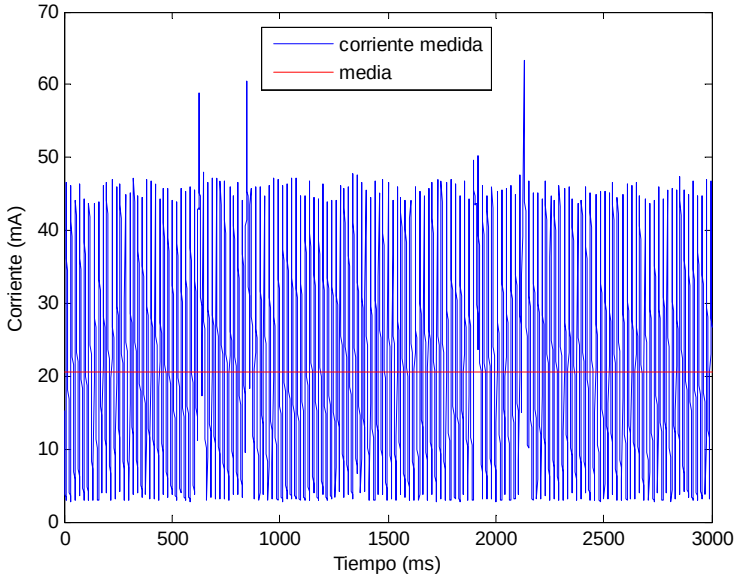


Figura 2.25: Consumo del dispositivo participando en una *piconet* como maestro

El consumo sostenido de *Bluetooth* en estado conectado dedicado al mantenimiento de la *piconet*, es pues notable incluso sin transferencia de datos, por lo que incorpora algunos modos de bajo consumo para reducir el consumo mediante el establecimiento de periodos de inactividad sin perder la sincronización entre el maestro y los esclavos.

2.4.3.3. Modos de bajo consumo

Los modos de bajo consumo de *Bluetooth* se explicaron con bastante nivel de detalle en la sección 2.2.4. El modo más simple es el modo *hold*, que corresponde a la captura representada en la figura 2.26. En este modo, los dispositivos maestro y esclavos acuerdan realizar de forma temporal una pausa de la actividad, lo que se refleja en una inmediata reducción del consumo a niveles similares a los de *standby* (en realidad algo mayor, unos 3mA de promedio, debido a que el dispositivo tiene que mantener la temporización). Una vez finalizado este periodo, que es transitorio, el dispositivo reanuda la actividad y vuelve a los niveles de consumo del modo conectado.

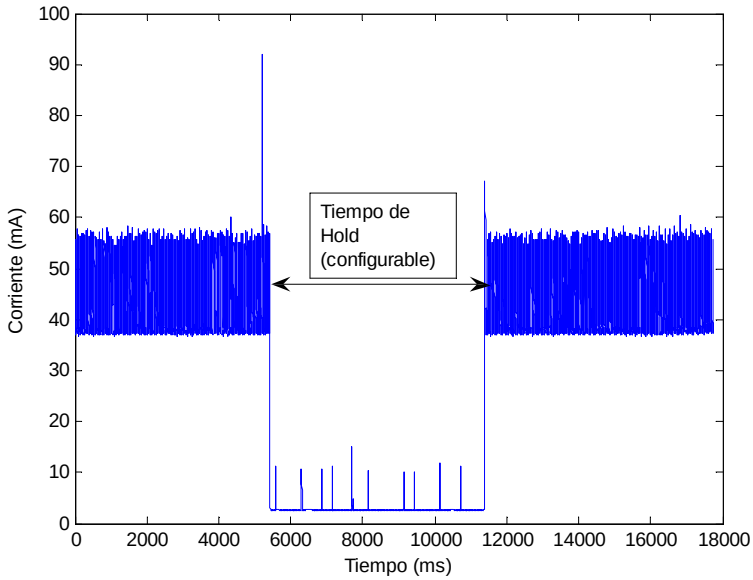


Figura 2.26: Consumo del dispositivo esclavo en modo *hold*

Los modos *sniff* y *park* no son transitorios, y el dispositivo permanece en ellos indefinidamente hasta que maestro y esclavo acuerdan volver al modo normal. En el modo *park* el dispositivo aparcado pierde su dirección de miembro de 3 bits, y no puede transmitir datos, despertándose de manera periódica únicamente con el objeto de mantener la sincronización con el maestro (y de esta forma evitar tener que hacer un nuevo proceso de *page* para reconectarse). Los consumos de maestro y esclavo se recogen en la figura 2.27 y 2.28. En este caso el consumo del maestro es mayor porque es el que transmite de forma regular los paquetes piloto.

En el modo *sniff*, el esclavo sigue permaneciendo como miembro activo de la *piconet*, pero negocia unas ventanas de actividad con su dispositivo maestro. El esclavo se despierta de forma regular con una periodicidad T_{sniff} y permanece despierto durante un intervalo

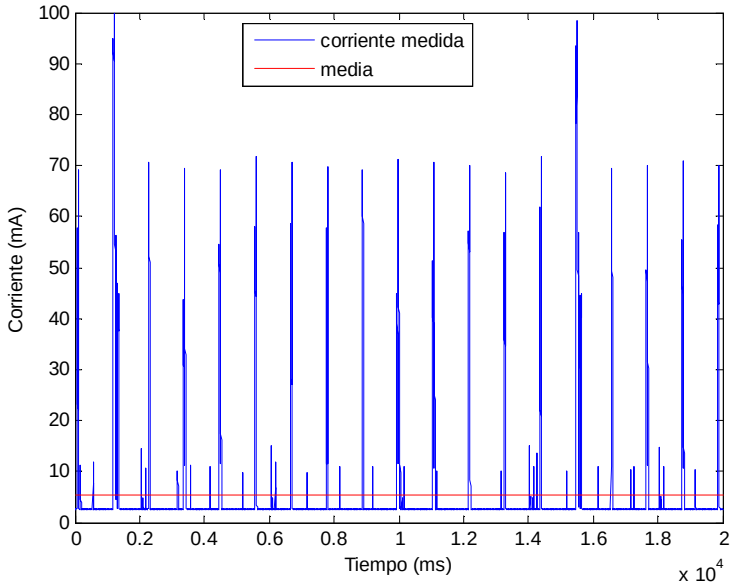


Figura 2.27: Consumo del dispositivo maestro con un único esclavo en modo *park*

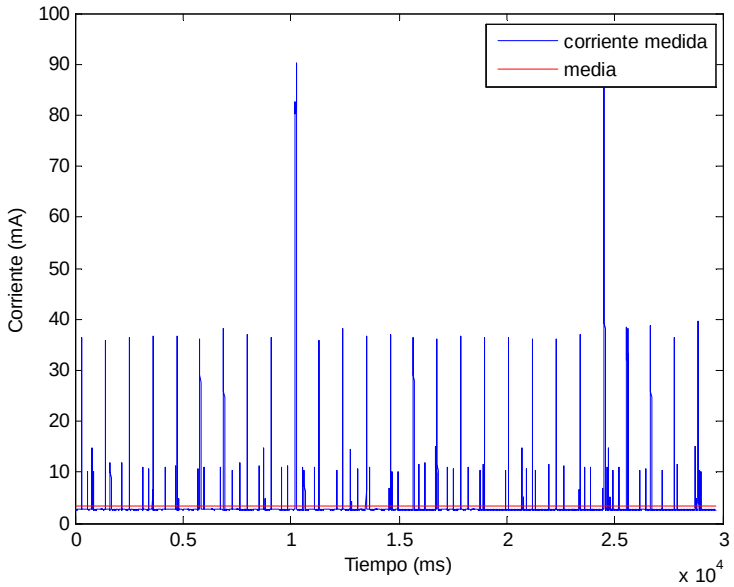


Figura 2.28: Consumo de un dispositivo esclavo en modo *park*

T_{win}, que se negocian con su maestro (en realidad se negocian el de *slots* (N_{sniff}) entre periodos y el número de *slots* $M/rigtharrowS$ ($N_{attempt}$)que el esclavo permanece activo esperando un sondeo por parte de su maestro, resultando $T_{sniff} = N_{sniff} \cdot T_{slot}$ y $T_{win} = N_{sniff} \cdot 2 \cdot T_{slot}$). En las figuras 2.29 y 2.31 se muestra respectivamente el comportamiento de esclavo y maestro para un periodo T_{sniff} de 1,5 segundos y una ventana T_{win} de 200ms. La figura 2.30 muestra también el comportamiento del esclavo, pero en este caso con $T_{win} = 20ms$.

Las líneas rojas muestran los valores promedio de consumo a lo largo de un periodo de 10 minutos. En este modo el consumo es aproximadamente proporcional al ciclo de trabajo es decir:

$$\overline{I_{sniff}} \approx \overline{I_{connected}} \cdot \frac{T_{win}}{T_{sniff}} + \overline{I_{idle}} \cdot \frac{T_{sniff} - T_{win}}{T_{sniff}} \tag{2.2}$$

Donde $\overline{I_{connected}}$ es el consumo medio del maestro o del esclavo cuando está participando activamente en la *piconet* (20mA para el maestro y 41 mA para el esclavo en el módulo evaluado), e $\overline{I_{idle}}$ es el consumo cuando desactiva la radio pero permanece sincronizado con la *piconet* (unos 3mA).

2.4.3.4. Compromiso entre prestaciones y consumo

Debido a que la utilización de los modos de bajo consumo puede suponer una degradación de las prestaciones, en [CCGC+09] se realizó también un análisis del efecto que pueden tener los diferentes parámetros del modo *sniff* en las prestaciones. En este caso se utilizó un módulo USB de Belkin con chipset de *broadcomm* conforme con el estándar 1.2 de *Bluetooth* y un sistema de pruebas similar al descrito más adelante en la sección 2.5.2 para permitir la generación de tráfico de diferente tipo y la medida de sus prestaciones.

Uno de los parámetros importantes del modo *sniff*, que no se ha mencionado hasta ahora en el estudio de consumo, es el parámetro $N_{timeout}$, que permite especificar cuánto *slots* prorroga el dispositivo su permanencia en modo activo tras recibir o transmitir un paquete durante la ventana de actividad. Lo más adecuado es poner este parámetro al menos a 1, para permitir que el dispositivo continúe activo y no se duerma mientras tenga datos que intercambiar, de forma que las prestaciones se vean lo menos alteradas posibles. En las figuras 2.32 y 2.33 se ilustra el comportamiento del comienzo de un flujo de datos sobre *Bluetooth* cuando $N_{timeout}$ es 0 y 1 respectivamente (la transmisión comienza alrededor del instante t=1s). El comportamiento antes de comenzar la transferencia de datos es el mismo y viene dado por los parámetros $N_{sniff} = 100$ y $N_{attempt} = 20$, sin embargo, una vez comenzada la transferencia del flujo de datos (simulada mediante una transferencia de un fichero por HTTP), el flujo de datos se transmite prácticamente sin interrupción cuando $N_{timeout} = 1$, mientras que con $N_{timeout} = 0$ se transmite únicamente durante las ventanas de *sniff*, penalizando bastante el retardo, y a la larga, también el consumo.

En las figuras 2.34 y 2.35 se muestra también el consumo energético del dispositivo esclavo al transmitir un flujo de datos utilizando diferentes tipos de paquete *Baseband*. La prueba se ha realizado tanto con el esclavo permanentemente activo (figura 2.35) como

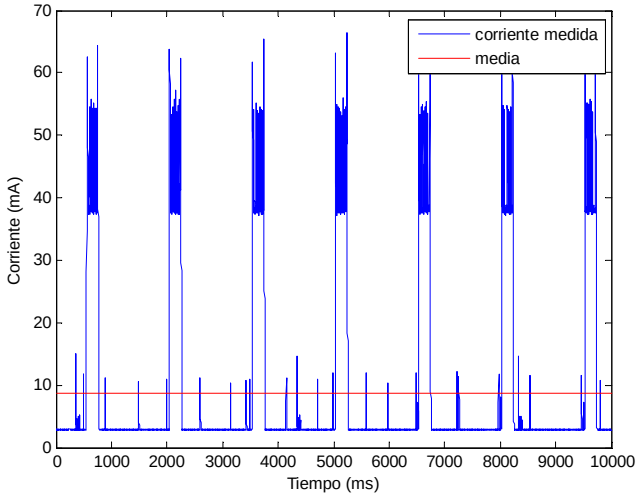


Figura 2.29: Consumo del dispositivo esclavo en modo *sniff* con $T_{sniff} = 1{,}5s$ $T_{win} = 200ms$

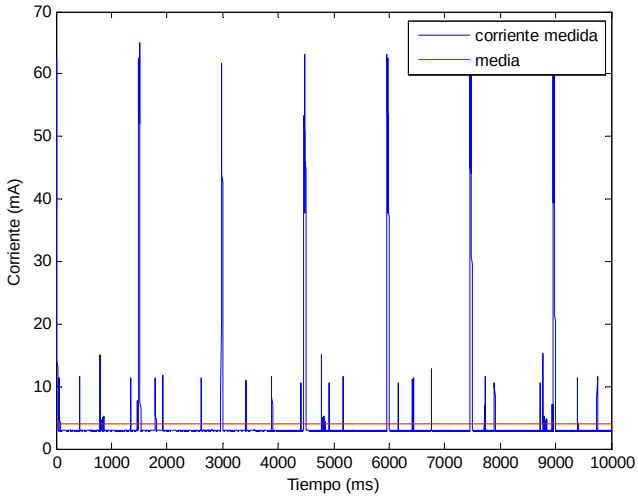


Figura 2.30: Consumo del dispositivo esclavo en modo *sniff* con $T_{sniff} = 1{,}5s$ $T_{win} = 20ms$

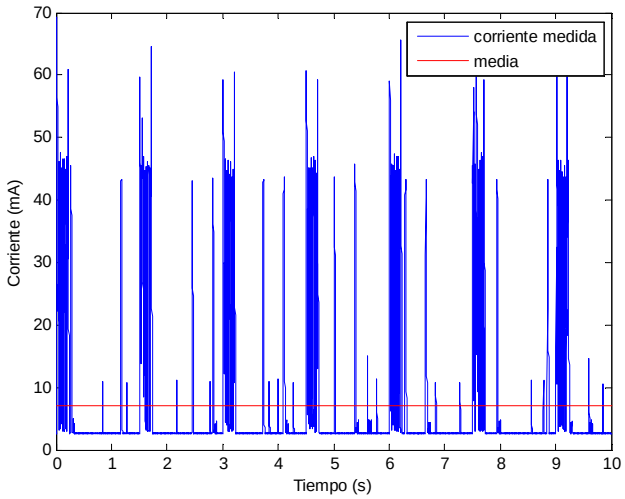


Figura 2.31: Consumo del dispositivo maestro en modo *sniff* con $T_{sniff} = 1{,}5s$ $T_{win} = 200ms$

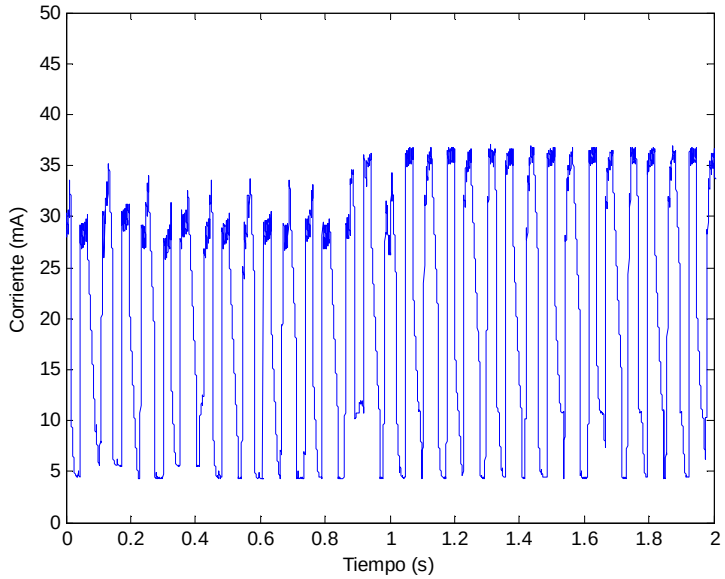


Figura 2.32: Consumo del dispositivo esclavo en modo *sniff* con $N_{timeout} = 0$

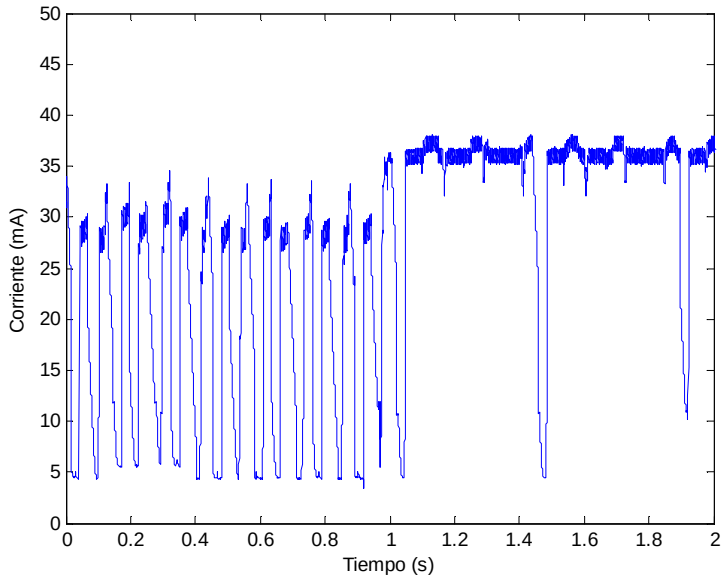


Figura 2.33: Consumo del dispositivo maestro en modo *sniff* con $N_{timeout} = 1$

	DM1	DM3	DM5	DH1	DH3	DH5
Maestro a esclavo, activo	98,4 kb/s	333,6 kb/s	410,4 kb/s	160,8 kb/s	460 kb/s	564 kb/s
Maestro a esclavo, *sniff*	80,8 kb/s	278,4 kb/s	283,2 kb/s	129,6 kb/s	401,6 kb/s	424,0 kb/s
Esclavo a maestro, activo	100 kb/s	336 kb/s	416,8 kb/s	154,4 kb/s	486,4 kb/s	577,6 kb/s
Esclavo a maestro, *sniff*	80 kb/s	272 kb/s	382,4 kb/s	128 kb/s	392 kb/s	520 kb/s

Tabla 2.1: *Throughput* medio para diferentes tipos de paquetes *Baseband* en modo activo y en modo *sniff* ($N_{sniff} = 20$, $N_{attempt} = 1$ y $N_{timeout} = 1$)

con el esclavo en modo *sniff* (figura 2.35) con $N_{sniff} = 20$, $N_{attempt} = 1$ y $N_{timeout} = 1$. Mediante comandos de la capa HCI enviados con la herramienta *hcitool* de *BlueZ* es posible habilitar o deshabilitar el uso de los diferentes tipos de paquetes *Baseband* que se emplean en el enlace ACL, permitiendo forzar que el envío se haga con un tipo concreto de paquete. En la prueba correspondiente a cada figura se ha realizado la transferencia mediante HTTP de un fichero de 512kB forzando en cada una el uso de uno de los tipos de paquete ACL (DM1, DM3, DM5, DH1, DH3 y DH5), para comprobar el efecto que tiene la elección del tipo de paquete en la energía gastada al transmitir un flujo de datos de igual tamaño. La línea azul representa el consumo instantáneo del dispositivo, y la línea azul un promediado que permite estimar el consumo medio. La tabla 2.1 recoge los valores promedio de *throughput* de un total de 10 conexiones en cada modo.

En los resultados se observa que la activación del modo *sniff* reduce apreciablemente el consumo del dispositivo en los periodos de inactividad, sin introducir una penalización excesiva en el *throughput* (alrededor del 20 % en la mayor parte de los casos) cuando hay información que transmitir gracias a que el parámetro $N_{attempt} > 0$ permite que el dispositivo permanezca activo casi todo el tiempo en dicho caso. Igualmente se aprecia que el consumo medio utilizando paquetes DM1 y DH1 es menor que utilizando paquetes multislot, lo cual es lógico ya que aunque la potencia de pico sea la misma en todos los casos, el dispositivo permanece más porcentaje del tiempo en transmisión cuando los paquetes son de mayor tamaño. Sin embargo, debido a que el tiempo necesario para transmitir la misma cantidad de información es más del doble, el consumo total de energía necesario para enviar la misma cantidad de datos es en realidad mayor. Un comportamiento similar se aprecia para el caso del dispositivo maestro, con la salvedad de que el consumo de éste durante la fase en la que no se están enviando datos es menor incluso aunque no esté en modo *sniff*.

Si en lugar de flujos más o menos continuos de transmisión el tráfico está conformado por paquetes que se transmiten de forma esporádica, como podría ser más habitual en una aplicación de baja actividad típica de una red de sensores, la activación del modo *sniff* sí puede tener un impacto importante sobre la latencia de dichos datos, ya que además de los retardos de transmisión existentes en el modo normal, hay que añadir el tiempo de espera a que el dispositivo entre en la ventana de *sniff* tras la llegada del paquete, que en media será $\frac{T_{sniff}}{2}$. Más adelante, en la sección 2.5.4 se muestran algunos resultados para diferentes

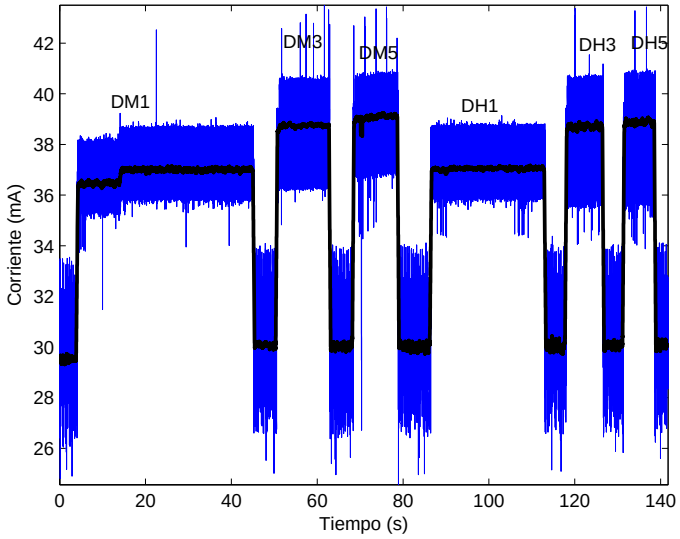


Figura 2.34: Consumo del dispositivo esclavo al transmitir un flujo de datos para distintos tipos de paquetes

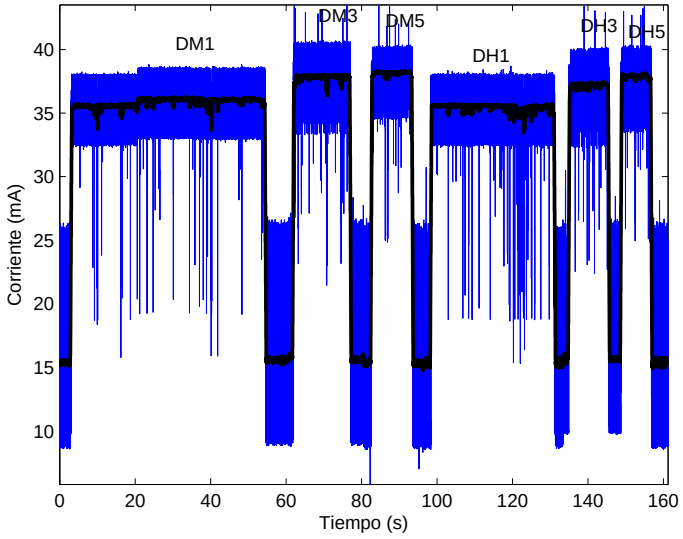
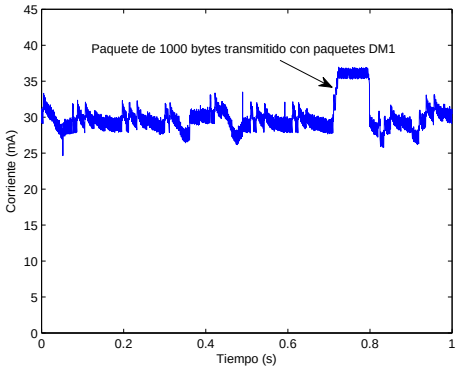
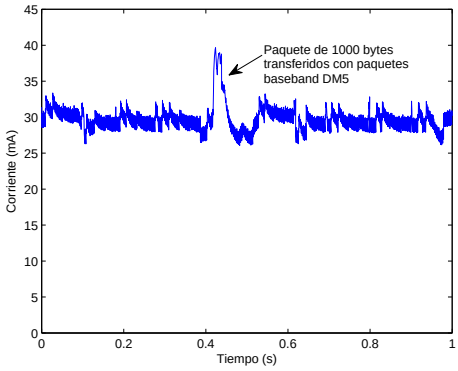


Figura 2.35: Consumo del dispositivo esclavo en modo *sniff* al transmitir un flujo de datos para distintos tipos de paquetes

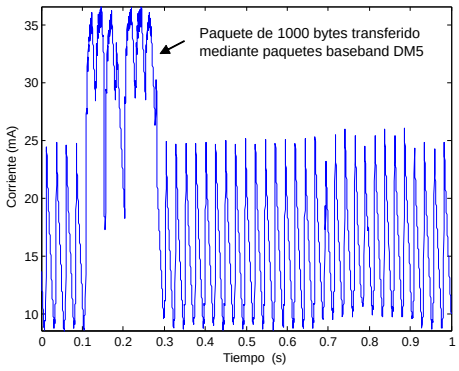
configuraciones de red, tanto *piconet* como *scatternet*. En lo que respecta al consumo y tiempo de transmisión (que no es lo mismo que el retardo total, ya que no incluye la espera a que el dispositivo se despierte), en la figura 2.36 se muestra el consumo de corriente del esclavo durante la transmisión de un paquete de 1000 bytes utilizando paquetes *Baseband* de tipo DM1 y DM5, en modo normal y en modo *sniff* (con los mismos parámetros antes mencionados). De nuevo, el consumo energético es menor para los paquetes DM5, ya que el ligero aumento de la corriente media se ve compensado por una menor duración de la transmisión. Así pues, la opción más adecuada desde el punto de vista del consumo parece ser la que habitualmente se sigue para maximizar la eficiencia del envío: escoger el tamaño del paquete menor que permita mandar de una sola vez todos los datos, o el máximo tamaño si no es posible enviar todos los datos en el mismo paquete *Baseband*.



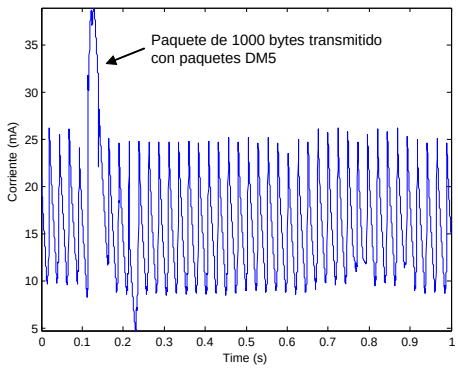
(a) Paquetes DM1



(b) Paquetes DM5



(c) Paquetes DM1, *sniff*



(d) Paquetes DM5, *sniff*

Figura 2.36: Consumo del dispositivo esclavo al transmitir un paquete con 1000 bytes de datos para distintos tipos de paquetes *Baseband*

2.4.3.5. Consumo del maestro en función del número de esclavos

También se han realizado pruebas para intentar caracterizar el consumo medio del maestro en función del número de esclavos presentes en la *piconet*, en ausencia de tráfico. Este estudio se ha realizado utilizando como maestros dos módulos diferentes: un *dongle* USB de OvisLink basado en el chipset *BlueCore4* de CSR, y otro dongle de Belkin con *chipset* de *Broadcomm*, ambos compatibles con la norma 2.1+EDR. Los resultados

absolutos se muestran en la figura 2.37, donde se observa que el consumo parece variar linealmente con el número de esclavos en ambos dispositivos, aunque los valores numéricos obtenidos son diferentes para cada uno de ellos, presentando el *chipset* de *Broadcomm* un consumo notablemente inferior (a pesar de que el consumo en estado «standby», no incluido en la figura, es superior para este módulo). Para comprobar la linealidad del consumo en ambos casos, se ha representado en la figura 2.38 el consumo normalizado, definido como:

$$Consumo_{Norm}(n_{esclavos}) = \frac{Consumo(n_{esclavos}) - Consumo(1)}{Consumo(7) - Consumo(1)} \tag{2.3}$$

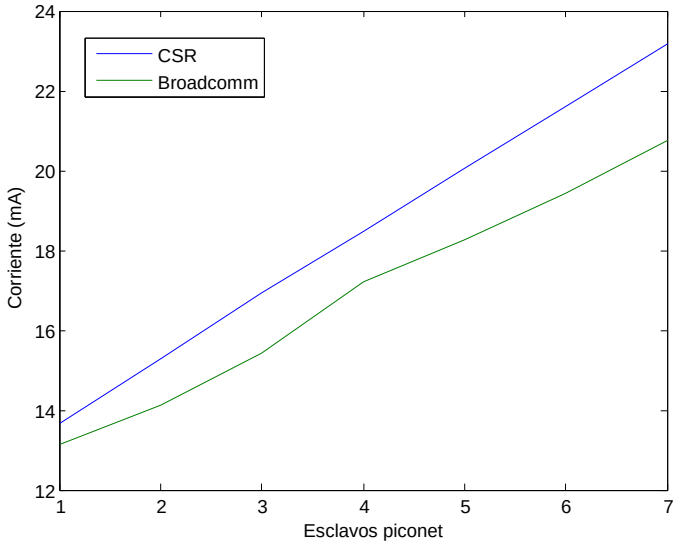


Figura 2.37: Consumo del maestro en función del número de esclavos

2.4.3.6. Consumo de un nodo participando en más de una *piconet* (*bridge*)

Finalmente se han realizado también medidas del consumo de un dispositivo que actúa como *bridge* en una *scatternet*, y que por tanto participa activamente en más de una *piconet*. En este caso se han utilizado los mismos dongles USB de ANSONIC utilizados más adelante en la sección 2.5, y que están basados de nuevo en el *Bluecore4* de CSR. El comportamiento de estos dispositivos para una *piconet* es similar a los anteriores, aunque su consumo de corriente en modo *standby* (con el módulo sin conectar y sin hacer ningún tipo de *scan*) es bastante elevado, de unos 10mA aproximadamente. Por lo demás, exhibe un comportamiento similar, consumiendo en media unos $14,5mA$ como maestro de un esclavo, unos $15,9mA$ como maestro de dos esclavos, y unos $21,3mA$ como esclavo. El consumo promedio como *bridge S/S* es en media prácticamente igual al medido en el rol de esclavo, mientras que como *bridge S/M* su consumo es algo mayor, de unos $22,7mA$. La figura 2.39 muestra capturas de la corriente instantanea medida para el dispositivo en los diferentes modos. En ella puede observarse que cuando se comporta como *bridge S/M* el dispositivo permanece la mayor parte del tiempo como esclavo, pero realiza regularmente

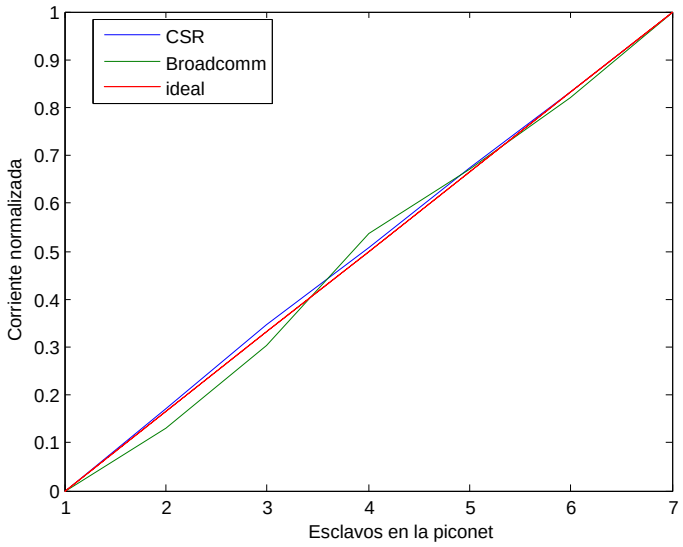


Figura 2.38: Consumo normalizado del maestro en función del número de esclavos

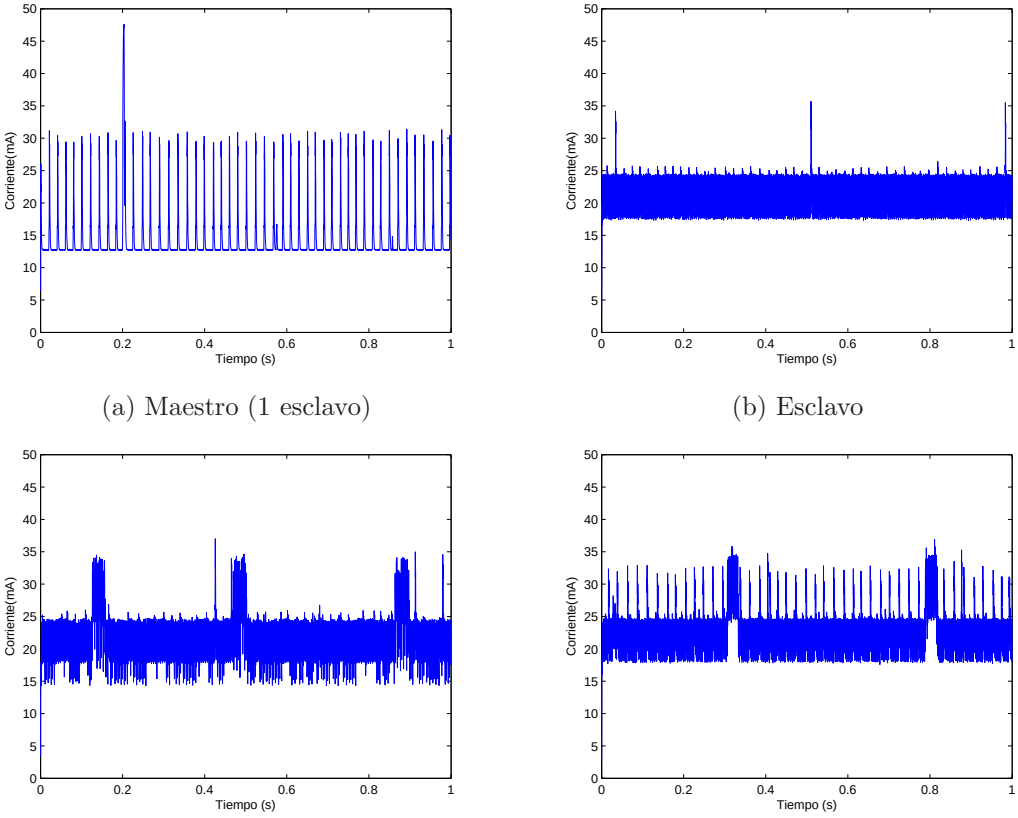
un sondeo a su dispositivo esclavo, igual que hace cuando se comporta únicamente como maestro.

También se ha estudiado el efecto del modo *sniff* en este escenario. La figura 2.40 muestra la corrriente instantánea de un dispositivo para $T_{sniff} = 1{,}5s$, $T_{win} = 40ms$ y $N_{timeout} = 1$ en ausencia de tráfico. Estos parámetros se han seleccionado para que se vea más claramente el comportamiento del dispositivo. Tal y como se observa, cuando el dispositivo está en modo *bridge* se despierta de forma periódica para participar en cada una de las *piconets* de las que es miembro, observándose dos ventanas de *sniff* separadas que se repiten periódicamente. En el caso del dispositivo *bridge S/M*, en una de ellas exhibe el comporamiento característico de un maestro, mientras que en la otra exhibe el comportamiento esclavo (la figura 2.41 muestra una ampliación con mayor detalle), en la que se visualizan dos ventanas de *sniff* casi consecutivas. En la primera el dispositivo actúa como esclavo y en la segunda, como maestro.

Se han realizado varias pruebas para determinar si las ventanas de *sniff* que se negocian aparecen en un orden concreto (por ejemplo siempre primero aquella en la que el dispositivo se encuentra en modo maestro y a continuación aquella en la que actúa como esclavo) y se ha encontrado que no es así. El orden de aparición, así como la separación entre las ventanas parece venir determinados por los *offsets* del reloj de los dispositivos, pudiendo darse tanto el orden mostrado en estas figuras como el contrario, y también una separación mayor entre las ventanas.

2.5. Aportaciones a la evaluación empírica de *scatternets*

Como se ha puesto de manifiesto en la sección 2.3.2, a pesar de la cantidad de estudios propuestos por los distintos autores, la viabilidad real de la formación de *scatternets* queda en entredicho porque en casi todos ellos se supone una serie de circunstancias no verificadas



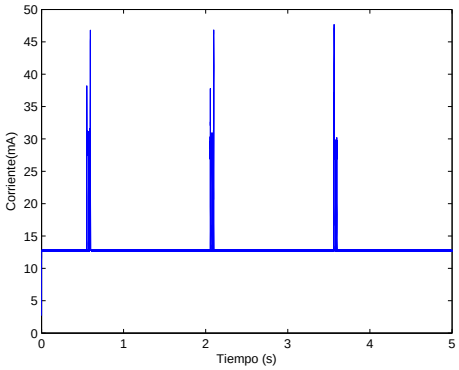
(a) Maestro (1 esclavo)

(b) Esclavo

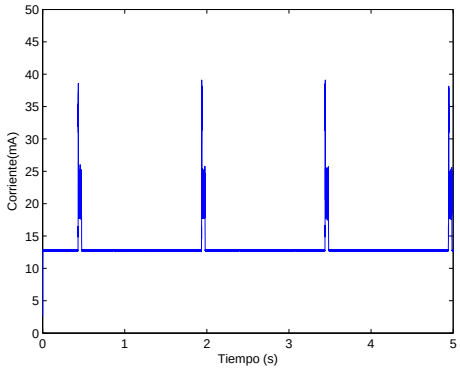
(c) *bridge S/S*

(d) *bridge S/M*

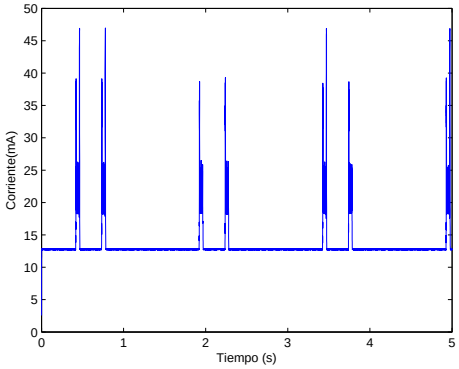
Figura 2.39: Consumo de un dispositivo como *bridge* en una *scatternet*



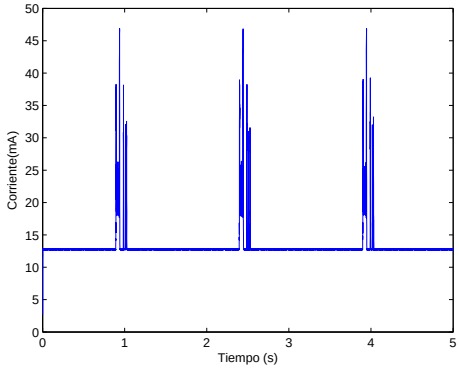
(a) Maestro (1 esclavo)



(b) Esclavo

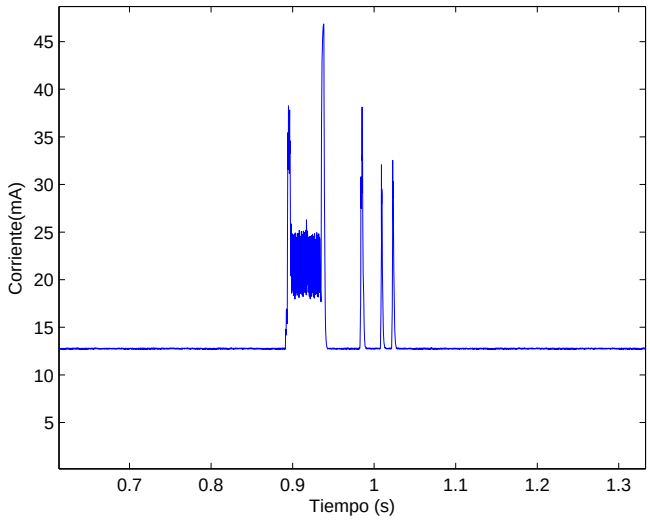


(c) *bridge S/S*



(d) *bridge S/M*

Figura 2.40: Consumo de un dispositivo como *bridge* en una *scatternet* en modo *sniff*



(a) *bridge S/M*

Figura 2.41: Detalle del consumo de un dispositivo *bridge S/M* durante la ventana de *sniff*

que deben cumplirse para que el estudio sea correcto. Los resultados se deducen en estos trabajos casi siempre en base a simulaciones o a estudios analíticos, que, debido a posibles inexactitudes o simplificaciones excesivas del modelo de funcionamiento, no tienen por qué coincidir con el comportamiento real que puede obtenerse si se implementan las distintas configuraciones en dispositivos reales.

En esta sección se incluyen pruebas que se han realizado sobre una plataforma de pruebas implementada con dispositivos comerciales, con objeto de investigar el comportamiento de las diferentes configuraciones de *bridges* que pueden darse al formar una *scatternet*, y determinar si puede obtenerse alguna mejora modificando alguno de los parámetros y modos de operación que pueden ser realmente configurados en tales dispositivos (tales como el tiempo de sondeo T_{Poll} o los parámetros de *sniff*).

Algunos de los resultados recogidos en esta sección han sido publicados en [CGCGP13].

2.5.1. Configuraciones básicas

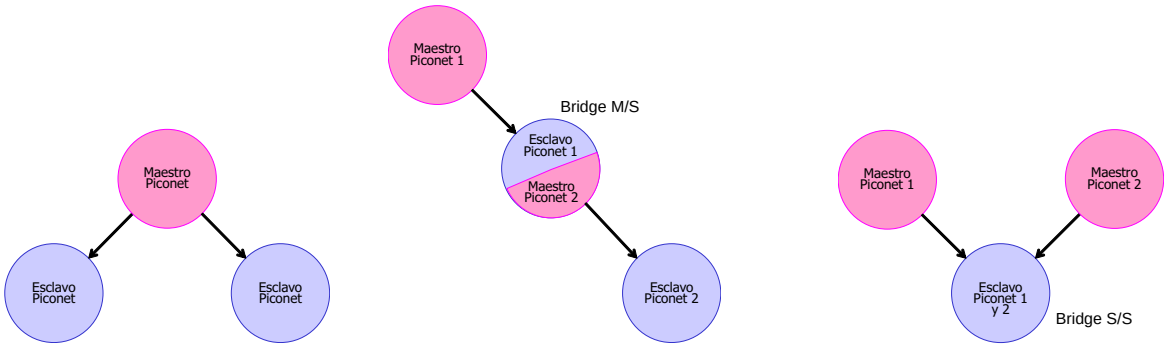
Como ya se ha mencionado previamente, dentro de las diferentes posibilidades de formación de puentes entre *piconets*, se pueden concretar dos subtipos básicos según los *roles* que tiene que adoptar el dispositivo *bridge*:

- Topología M-S/M-S: En esta topología el dispositivo puente actúa como como maestro de una de las *piconets* y como esclavo de la otra (dipositivo puente de tipo maestro-esclavo). A este tipo de dispositivo puente se le suele denominar *bridge S/M*
- Topología M-S-M: En esta topología, el dispositivo puente participa en ambas *piconets* como esclavo (dispositivo puente de tipo *bridge S/S*).

Los dispositivos puentes de tipo S/S podrían emplearse para unir más de dos *piconets*, ya que el dispositivo puente podría participar como esclavo en más de dos de ellas. La norma no establece ninguna limitación en este sentido, aunque debido a los recursos necesarios para mantener la sincronización con diferentes maestros, es de suponer que tal limitación exista en una implementación real (de hecho, en las pruebas realizadas se ha encontrado que los dispositivos comerciales sólo aceptan normalmente conectarse simultáneamente a dos maestros como mucho). Sólo algunos de los algoritmos propuestos en los trabajos citados en la sección 2.3.2 consideraba este parámetro limitado a dos, mientras que en algunos otros el número medio de maestros al que tiene que conectarse un *Brige S/S* se considera una de las métricas a evaluar (mejor mientras menor sea el valor medio).

Dada esta diferenciación, se han evaluado tres tipos de escenarios básicos que se muestran en la figura 2.42. El escenario «I» corresponde a una *piconet* y se utiliza como marco de referencia para la evaluación de los otros dos escenarios, de forma que permita comprobar el impacto negativo que tiene la configuración *scatternet* sobre las prestaciones. El escenario «II» no es simétrico y presenta a su vez dos casos distintos para ser evaluados: las comunicaciones entre el nodo Maestro *puro* y su esclavo el *bridge S/M*, y las comunicaciones entre éste último y su esclavo. El escenario «III» sí es simétrico y por tanto sólo se estudia uno de los enlaces, asumiendo que el comportamiento del otro será similar.

Además de estos escenarios básicos, se han estudiado otros algo más complejos, que se describirán más adelante.



I) *Piconet* II) *Scatternet* con *bridge S/M* III) *Scatternet* con *bridge S/S*

Figura 2.42: Configuraciones básicas evaluadas

2.5.2. Descripción del sistema de prueba

El escenario de pruebas se muestra en la figura 2.43, y está formado por tres ordenadores personales (PCs) con conexión a una red de área local Ethernet y equipado cada uno con un dongle *Bluetooth* USB. Todas las pruebas mostradas en esta sección se han realizado con *dongles Bluetooth* USB clase 2 comerciales basados en un chipset BlueCore 4 de CSR y compatibles con el estándar *Bluetooth* 2.1+EDR. También se han realizado pruebas puntuales con otros *dongles* USB con chipsep de *Broadcomm* compatibles con el mismo estándar, y los resultados obtenidos en tales casos han sido similares, con excepción de algunos problemas de estabilidad que presentan los módulos de broadcomm tras un periodo de operación prolongado y que pueden ser achacables a la presencia de algún bug en su *firmware*.

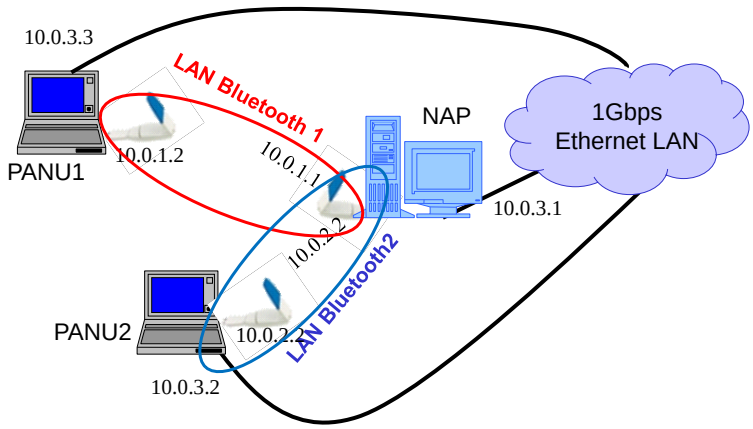


Figura 2.43: Sistema de evaluación de topologías *scatternet* básicas

Las pruebas se han realizado aprovechando las herramientas de código abierto disponibles en la pila de protocolos *Bluetooth* para Linux, *BlueZ*, que han sido modificadas para añadirles soporte para determinadas acciones que son soportadas por el módulo mediante

comandos HCI pero que no eran ofrecidas por las herramientas (por ejemplo la configuración de los modos de bajo consumo, como *sniff* y sus parámetros). Así mismo ha sido necesario parchear y recompilar los módulos de *Bluetooth* del *kernel* de Linux para evitar que la propia pila de protocolos cancele automáticamente el modo *sniff* en una conexión al detectar la presencia de tráfico en ella.

Para la realización de las pruebas se ha utilizado el perfil de red de área local (PAN) de Bluetooth, que utiliza el protocolo de encapsulamiento BNEP para permitir el transporte de paquetes IP por *Bluetooth*. Este perfil presenta las siguientes ventajas dentro del marco de las pruebas que se pretenden realizar:

- El encaminamiento de la información lo resuelve IP, dentro del nivel de ejecución del *kernel* del sistema operativo y no tiene que realizarlo la aplicación.
- Es posible aprovechar herramientas de monitorización y generación de tráfico ya disponibles (por ejemplo ping, hping2, tcpdump, etc.)
- Se facilita la rápida creación de herramientas de medida personalizadas utilizando las API *POSIX* y *Socket* de Linux.

Mediante las herramientas de BlueZ (antiguamente mediante la herramienta *pand* y actualmente mediante un *script* python que interacciona con BlueZ a través del subsistema DBus), uno de los PCs se configura como dispositivo punto de acceso *Bluetooth* (NAP) y los otros dos se configuran como dispositivos clientes (PANU). Cada uno de los clientes PANU establece una red de área local con el dispositivo NAP, que es por tanto el que va a actuar de puente. Como puede observarse en la figura 2.43, cada uno de los dispositivos PANU dispone de dos interfaces de red, uno de los cuales corresponde a una conexión de área local *Bluetooth* con el NAP (bnep0) y el otro a la conexión Ethernet (eth0). Cada uno de estos interfaces está configurado con una dirección IP distinta, dando lugar a dos subredes separadas a las que pertenece el dispositivo. El NAP por su parte dispone de 3 interfaces de red, uno de ellos corresponde a la red Ethernet (eth0) y los otros dos (bnep0 y bnep1) a las conexiones establecidas con los PANUs. De nuevo, a cada interfaz se le asigna una dirección IP de forma que el dispositivo pertenece a tres subredes IP simultáneamente.

Al realizar por primera vez la conexión de los PANUs al NAP se formará una *piconet* en la que el NAP actuará como maestro y los PANUs como esclavos. No obstante, esta configuración puede ser modificada en cualquier momento utilizando la herramienta *hcitool* de *BlueZ* para ordenar un cambio de rol a través del procedimiento *Role-Switch* de *Bluetooth*. Al ordenar un cambio de rol del NAP con uno de los PANUs, se forma una *scatternet* en la que el NAP toma el papel de *bridge S/M*, mientras que si el cambio de role se hace con ambos PANUs, el NAP se comporta como *bridge S/S*.

El objetivo del sistema de prueba es medir el retardo de los paquetes que viajan en sentido ascendente y descendente entre el NAP y cada uno de los PANUs para las diferentes configuraciones del NAP como *bridge* de la *scatternet*. Para poder medir el tiempo de ida o vuelta (lo que en la literatura se define normalmente como OTT, *One Trip Time*), el sistema de prueba aprovecha que el NAP y los PANUs están también conectados mediante una red Ethernet de alta velocidad. Como se muestra en la figura

2.44, cada PANU se configura de forma que los paquetes recibidos desde el NAP por el interfaz *Bluetooth* se reflejen de nuevo hacia el NAP por el interfaz Ethernet, y viceversa, los paquetes recibidos por el interfaz Ethernet, se redireccionarán hacia el NAP por el interfaz Bluetooth. Para conseguir esto, se habilita el enrutamiento IP en los PANUs, y se hace uso de la herramienta *iptables* del sistema *Netfilter* de *Linux* para configurar un conjunto de reglas de NAT (*Network Address Translation*) estático.

De esta forma un paquete IP inyectado por una aplicación de medida que se ejecute en el NAP hacia el interfaz Bluetooth, será recibido por su interfaz Ethernet, permitiendo estimar el retardo de *Bluetooth* en sentido ascendente, y los paquetes enviados por la aplicación de medida en el NAP hacia el PANU por el interfaz Ethernet, serán recibidos por el correspondiente interfaz *Bluetooth* del NAP, permitiendo medir el retardo de *Bluetooth* en sentido descendente (recepción). Esto permite que las medidas sean realizadas por un único PC y por tanto no es necesario que los relojes de los diferentes equipos que forman la red de pruebas estén sincronizados, simplificando mucho el desarrollo de la herramienta de medida, que se ejecuta únicamente en el NAP. Obviamente, en todo este proceso se está despreciando el retardo de los paquetes en el tramo Ethernet (que normalmente es inferior a 0.1 ms) frente al que experimentan en el tramo *Bluetooth*. A su vez, la ventaja de reflejar los paquetes mediante reglas de NAT aplicadas en pre y post-enturamiento, en lugar de hacerlo mediante una aplicación residente en el PANU, es que este tratamiento se hace a nivel del *kernel* de *Linux* y por tanto en principio con menos retraso añadido. Por último el hecho de que la aplicación de prueba se ejecute en el NAP y éste sea el que controle la topología de la red *Bluetooth*, facilita la ejecución automatizada de las pruebas.

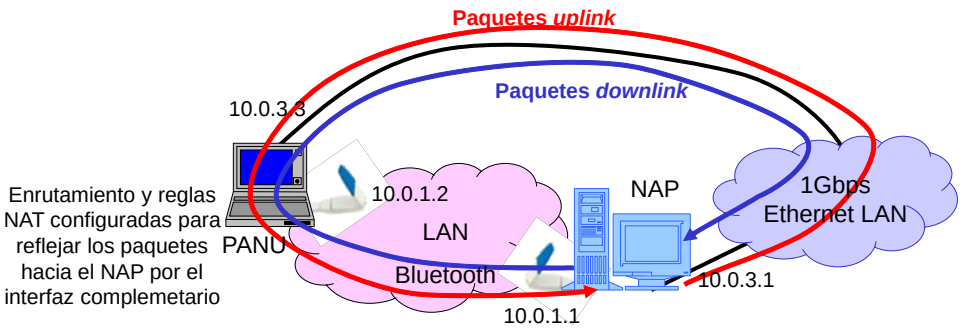


Figura 2.44: Sistema de evaluación de topologías *scatternet* básicas

Una opción evidente que se estuvo barajando para la realización de las pruebas era conectar todos los *dongles* USB al mismo PC y utilizar conexiones RFCOMM utilizando la correspondiente API de *BlueZ*, pero se desestimó al detectar experimentalmente que la pila de protocolos de *BlueZ*, que implementa los niveles superiores desde L2CAP hacia arriba, no gestiona bien las conexiones entre varios interfaces *Bluetooth* del mismo equipo (lo cual es lógico, ya que no tiene mucho sentido, al margen de la realización de pruebas, que un equipo se conecte consigo mismo).

Para la realización de las medidas se desarrolló una herramienta que envía paquetes UDP de prueba de forma independiente (no sincronizada) en ambos sentidos de transmisión. Cada paquete se marca con un número de secuencia y una marca de tiempo

(*timestamp*) que viajan en el campo de datos UDP. El paquete es reflejado por el PANU y enviado por el interfaz complementario, volviendo de nuevo al NAP, donde es recibido y procesado por la herramienta, que puede calcular el tiempo de ida y vuelta (RTT, *Round Trip Time*) mediante la marca de tiempo. Si se considera despreciable el tiempo de viaje por el interfaz *ethernet* frente al retardo del enlace *Bluetooth* (lo cual se ha verificado), los tiempos medidos se puede considerar una buena estimación del retardo OTT en el interfaz *Bluetooth*.

En la realización de algunas pruebas preliminares con una primera versión de la herramienta de medida, que enviaba paquetes de forma totalmente periódica, se detectaron algunas distorsiones en las medidas debidas a efectos de sincronización:

- Al enviarse el tráfico de forma simultánea en ambos sentidos, la existencia de tráfico en sentido descendente beneficiaba al sentido ascendente, ya que al realizar el envío en sentido descendente el maestro comprueba si el esclavo tiene información para él de forma inmediata.

- Al enviarse el tráfico de forma totalmente periódica, se produce un cierto efecto de sincronización o de deriva constante cuando se introducen otros comportamientos de naturaleza periódica en la red, como es el caso del modo *sniff*. Esto provoca una oscilación periódica en el retardo que afecta a las estadísticas extraídas.

Para evitar estas distorsiones en las medidas (*artifacts* es el término utilizado comúnmente en la literatura en inglés para estos fenómenos), la herramienta empleada permite el envío de tráfico de forma aleatoria, y el proceso de envío es independiente para cada sentido. El tiempo entre paquetes es aleatorio, con una distribución de probabilidad uniforme cuyo máximo y mínimo valor se pueden configurar en cada prueba. En cada ejecución también puede configurarse el tamaño de los paquetes enviados y si se inyecta o no tráfico en cada sentido.

2.5.3. Resultados para las configuraciones básicas

Para establecer un primer escenario de referencia, se ha realizado un conjunto de pruebas enviando paquetes de pequeño tamaño (20 bytes de datos a nivel UDP a los que habrá que sumar los 28 bytes de sobrecarga que impone la cabecera UDP/IP) en un único sentido, con un intervalo medio entre envíos de 100 ms enviados siguiendo una distribución de probabilidad uniforme de 80 a 120 ms (es decir, tras el envio de cada paquete se espera un tiempo aleatorio entre 20 y 120ms para enviar el siguiente). Las medidas en cada posible escenario están basadas en una serie de 10.000 paquetes enviados en un único sentido de transmisión, lo que correspondería a un escenario en el que no otro hay tráfico de fondo. Los parámetros de calidad de servicio que afectan al funcionamiento del nivel *link-manager* están configurados a los valores por defecto, lo cual determina, según la documentación del fabricante del *chipset* del *dongle* que el tiempo T_{Poll} está configurado a 40 *slots* (25ms).

La figura 2.45 muestra los histogramas del retardo de los paquetes calculado a partir de la muestra de 10.000 paquetes de cada escenario. 2.45.a y 2.45.b contienen las estadísticas correspondientes a los enlace *downlink* ($M \rightarrow S$) y *uplink* ($M \leftarrow S$), respectivamente, de

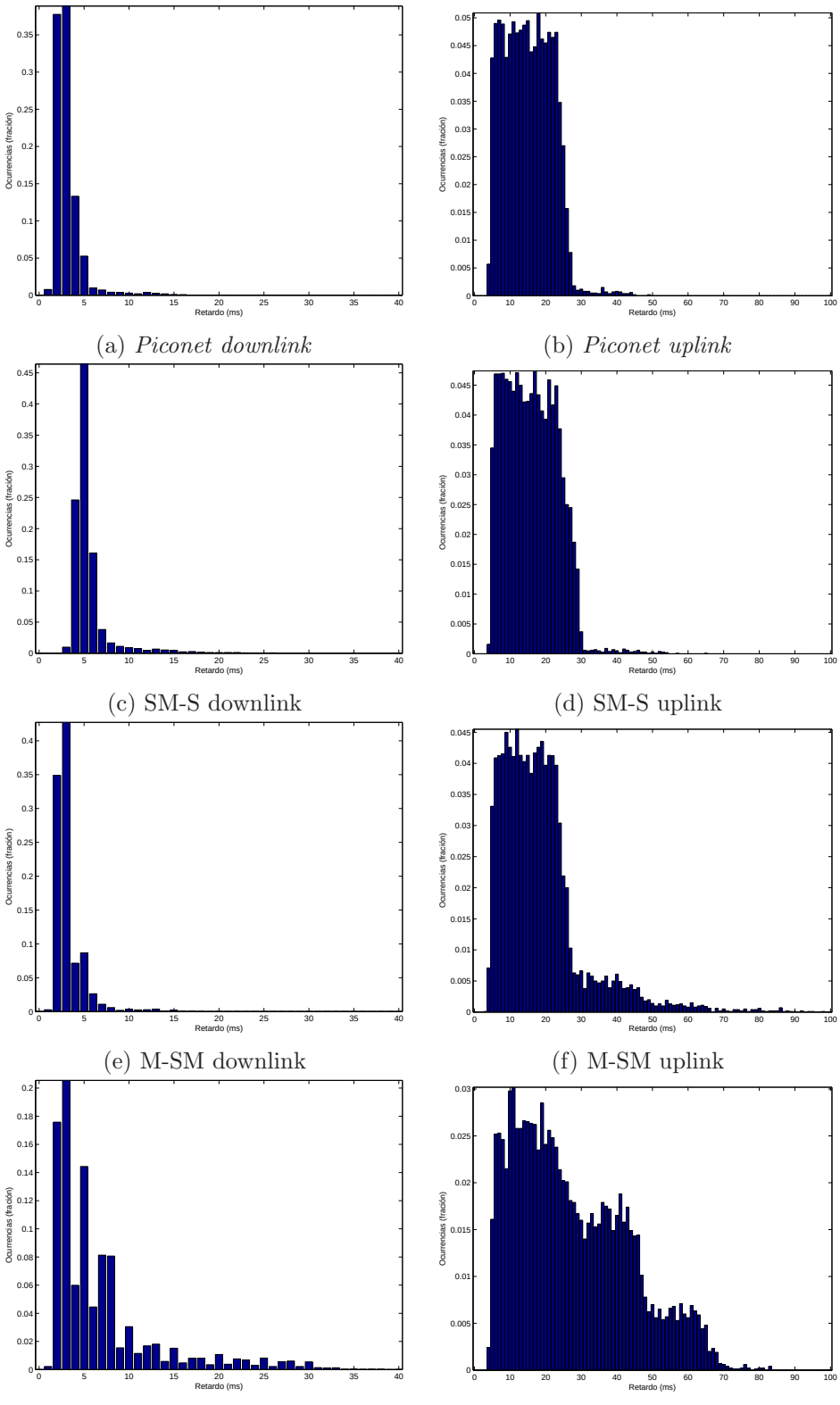
Parámetro (ms)/Escenario		$M \leftrightarrow S$	$S/M \leftrightarrow S$	$M \leftrightarrow S/M$	$M \leftrightarrow SS$
Downlink	Media	3,54	6,03	3,68	7,31
	Mediana	3,10	5,40	3,20	5,30
Uplink	Media	15,61	16,55	19,05	27,98
	Mediana	15,40	16,20	17,00	24,80

Tabla 2.2: Estadísticas tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 80 y 120 ms

una *piconet* con un maestro y dos esclavos. Las figuras 2.45.c-f muestran los resultados correspondientes a una *scatternet* con *bridge S/M*, donde 2.45.c-d corresponden al enlace entre el *bridge S/M* y su esclavo, y las figuras 2.45.e-f corresponden al enlace entre el *bridge S/M* y su maestro. Por tráfico *downlink* (figuras 2.45.c y 2.45.e) se entiende el sentido de comunicación que va desde el maestro hacia el esclavo de cada enlace. Igualmente, el sentido *uplink* (subfiguras d y f) corresponde al tráfico que va desde el dispositivo que actúa como esclavo al que actúa como maestro en cada enlace. Finalmente las subfiguras g y h contienen los resultados para uno de los enlaces de un *bridge S/S*, manteniendo los mismos criterios para distinguir el sentido del tráfico. Además, la tabla 2.2, muestra el valor medio y la mediana del retardo de los paquetes medido en cada uno de los escenarios.

Como puede observarse, el retardo de los paquetes en sentido *downlink* es en valor medio inferior y está sometido a una menor dispersión que en sentido *uplink*, lo cual es lógico, ya que el maestro puede decidir enviar los datos hacia el esclavo en cuanto estén disponibles, mientras que el esclavo tiene que esperar a ser sondeado por el maestro. De hecho el histograma del retardo en sentido *uplink* en la *piconet* se ajusta bastante a una distribución uniforme de 25ms de intervalo, justo el tiempo T_{poll} configurado por defecto, que es lo que cabría esperar cuando los paquetes llegan de forma incorrelada con el proceso de sondeo.

El comportamiento de los enlaces en la *scatternet* con *bridge* híbrido S/M es bastante parecido al de la *piconet*, con algunas excepciones. Puesto que el *bridge* híbrido sigue sondeando a su dispositivo esclavo con la periodicidad marcada por T_{poll}, el retardo en sentido $S/M \leftarrow S$ (subfigura d) es muy similar al de la *piconet*. El retardo en sentido $M \leftarrow S/M$ (subfigura f) se ve en cambio algo perjudicado, debido a que puesto que el nodo *bridge* está obligado a sondear a sus esclavos cada cierto tiempo, por lo que existe cierta probabilidad de perder a su vez el paquete de *POLL* enviado por su dispositivo *padre* maestro, teniendo que esperar un segundo periodo de *poll* a ser sondeado de nuevo, de ahí la *cola* que se observa en el histograma. En cualquier caso los resultados parecen indicar que dicha pérdida no es muy frecuente, por lo que se deduce que el dispositivo *bridge S/M* permanece la mayor parte del tiempo como esclavo en la *piconet* de su dispositivo padre, y sólo cuando es necesario (para realizar el sondeo o enviar un paquete) la abandona y pasa a su propia *piconet*. Esto también explica la existencia de un retardo algo mayor en el sentido downlink en el enlace $S/M \rightarrow S$ (subfigura c), que se explicaría porque tras la llegada del paquete el *bridge* tiene que cambiar de *piconet* y de *rol* y conmutar la secuencia de saltos de frecuencia.



(a) *Piconet downlink* (b) *Piconet uplink*

(c) SM-S downlink (d) SM-S uplink

(e) M-SM downlink (f) M-SM uplink

(g) SS downlink (h) SS uplink

Figura 2.45: Resultados tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 80 y 120 ms

La mayor diferencia con la *piconet* se produce en el caso del *bridge S/S* (subfiguras g y h). En este caso el comportamiento parece indicar que el dispositivo esclavo está conmutando casi contínuamente entre las dos *piconets*, existiendo una probabilidad aproximada del 50 % de no recibir el paquete *POLL* de cada uno de sus maestros. Como puede observarse, esto se refleja en que la distribución de probabilidad del retardo uplink $M \leftarrow S/S$, aproximada por el histograma (figura h), parece volverse multimodal, apareciendo varios intervalos que corresponderían a aquellos paquetes que tienen que esperar a un nuevo sondeo tras la pérdida de uno o incluso varios paquetes *POLL*.

Estos resultados son coherentes con las observaciones y el modelo propuesto por [LG10] para una *piconet*. El modelo básico de retardo propuesto en dicha tesis permite calcular el retardo medio *uplink* y *downlink* para paquetes pequeños (sin fragmentación) y en ausencia de pérdidas mediante las ecuaciones 2.4 y 2.5, respectivamente:

$$\overline{t_{down_{pico}}} = 3 \cdot T_{slot} + t_{tx} + C_{usb} \tag{2.4}$$

$$\overline{t_{up_{pico}}} = 2 \cdot T_{slot} + \frac{T_{poll}}{2} + t_{tx} + C_{usb} \tag{2.5}$$

Donde T_{slot} es la duración de un *slot*, T_{poll} el tiempo de sondeo, t_{tx} el tiempo de transmisión del paquete radio, y C_{usb} el tiempo de transporte del paquete por los interfaces USB entre cada *dongle* y su PC, siendo los dos últimos términos dependientes del tamaño del paquete. Los valores medios obtenidos para la *piconet* son compatibles con esta fórmula, pero además podemos modificar el modelo *uplink* para la *scatternet* con *bridge S/S* considerando la probabilidad de pérdida de paquetes de *POLL*. En este caso, el periodo efectivo de sondeo podría verse aumentado debido a las pérdidas:

$$\overline{t_{up_{s/s}}} = 2 \cdot T_{slot} + \frac{(1-p) \cdot T_{poll} + p \cdot (1-p) \cdot 2 \cdot T_{poll} + p^2 \cdot (1-p) \cdot 3 \cdot T_{poll} + ...}{2} + t_{tx} + C_{usb} \tag{2.6}$$

$$\overline{t_{up_{s/s}}} = 2 \cdot T_{slot} + \frac{T_{poll}}{2} + p \cdot \frac{T_{poll}}{2} + p^2 \cdot \frac{T_{poll}}{2} + ... + t_{tx} + C_{usb} \tag{2.7}$$

$$\overline{t_{up_{s/s}}} = 2 \cdot T_{slot} + \frac{1}{1-p} \cdot \frac{T_{poll}}{2} + t_{tx} + C_{usb} \tag{2.8}$$

Donde p es la probabilidad de perder un paquete de poll, que es 0,5 en el caso de la *scatternet* S/S, lo que resultaría en la expresión 2.9, que es coherente con los valores medios que figuran en la tabla 2.2. En realidad no es esperable que el paquete se mantenga indefinidamente el el *buffer*, por lo que la diferencia entre los retardos medios en realidad debería ser ligeramente inferior a $T_{poll}/2$.

$$\overline{t_{up_{s/s}}} = 2 \cdot T_{slot} + T_{poll} + t_{tx} + C_{usb} = t_{up_{pico}} + \frac{T_{poll}}{2} \tag{2.9}$$

En el sentido *downlink* de este mismo escenario, se puede seguir un razonamiento similar, con la salvedad de que en este caso lo que el esclavo deja de recibir no es el paquete

de *POLL* sino el paquete con los datos, por lo que éste tendrá que ser retransmitido de nuevo por el dispositivo maestro. En este caso, podría aplicarse una versión simple del modelo derivado por [LG10] para el retardo *downlink* en una *piconet* en presencia de retransmisiones por pérdidas de paquete, que viene dada por 2.10:

$$\overline{t_{down_{s/s}}} = 3 \cdot T_{slot} + \frac{p}{(1-p)} \cdot \overline{t_{rtx}} + t_{tx} + C_{usb} = t_{down_{pico}} + \frac{p}{(1-p)} \cdot \overline{t_{rtx}} \qquad (2.10)$$

Donde $\overline{t_{rtx}}$ es el tiempo medio empleado en retransmitir un paquete, y que en principio puede depender de diversos factores como el número de esclavos que el maestro tiene que atender. Si asumimos que la probabilidad de pérdida es de 0,5 debido a que el nodo *bridge* reparte el tiempo entre sus dos maestros, entonces obtenemos que $\overline{t_{down_{s/s}}} = \overline{t_{down_{pico}}} + \overline{t_{rtx}}$. A través de observaciones en diferentes pruebas realizadas, en [LG10] se constata empíricamente que el tiempo mínimo entre dos direccionamientos del maestro a un mismo esclavo es de 6 *slots* (3,75 ms), que se aproxima bastante a la diferencia entre las medias estimadas del retardos de los paquetes en el enlace *downlink* de los escenarios *piconet* y *scatternet* con *bridge S/S*.

Para confirmar este comportamiento, se ha realizado una batería de pruebas adicional modificando los parámetros de calidad de servicio. Según la documentación del fabricante del *chipset* de los módulos *Bluetooth* utilizados, el tiempo T_{poll} se puede variar mediante el comando HCI para establecer la calidad de servicio. La mayor parte de los parámetros de este comando (*token-rate*,*peak-bandwidth*) son ignorados explícitamente, pero el tiempo T_{poll} se ajusta de forma aproximada al valor *latency* indicado en el comando. La figura 2.46 muestra el retardo medido en sentido *uplink* entre el *bridge S/S* y uno de sus maestros, en función del parámetro *latency* configurado en el comando de calidad de servicio mediante la versión modificada que se ha desarrollado de la herramienta *hcitool* de *BlueZ*. Los puntos rojos corresponden a los valores medidos, y la línea azul es una recta con pendiente unidad. Teniendo en cuenta que la documentación del fabricante sólo garantiza el ajuste aproximado del tiempo T_{poll} al valor *latency* especificado, indicando que no todos son posibles, los datos obtenidos sí parecen confirmar la validez del modelo propuesto.

Estas expresiones aproximadas podrían utilizarse también para determinar las prestaciones en un hipotético *bridge S/S* con N dipositivos maestros, suponiendo una probabilidad de pérdida de $p = (N-1)/N$. Aunque se proponen con cierta frecuencia en la literatura relacionada con los algoritmos de formación, en la práctica no hemos encontrado ningún dispositivo de electrónica de consumo que permitiera conexiones con más de dos maestros manteniendo un comportamiento estable. No obstante, es posible que existan algunos dispositivos especializados que sí lo permitan, por ejemplo los responsables del desarrollo *hardware* del módulo BtNode descrito en [Beu05a] señalan que el controlador *Bluetooth* que utiliza su diseño está basado en un módulo con un firmware especial, el *Zeevo ZV4002*, cuyas especificaciones indican que permite hasta 7 esclavos y 3 maestros (en cualquier caso, la propuesta posteriormente recogida por sus autores en [BDMT05] se orienta a la formación de redes en árbol utilizando únicamente *bridges S/M*, que sí es soportada por un gran número de dispositivos comerciales).

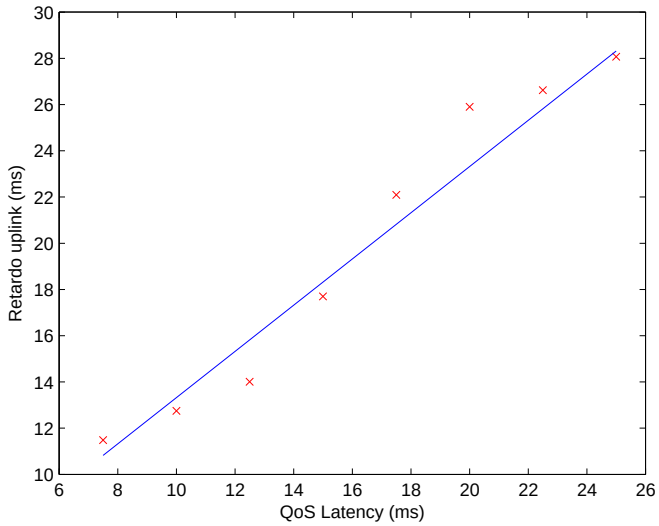


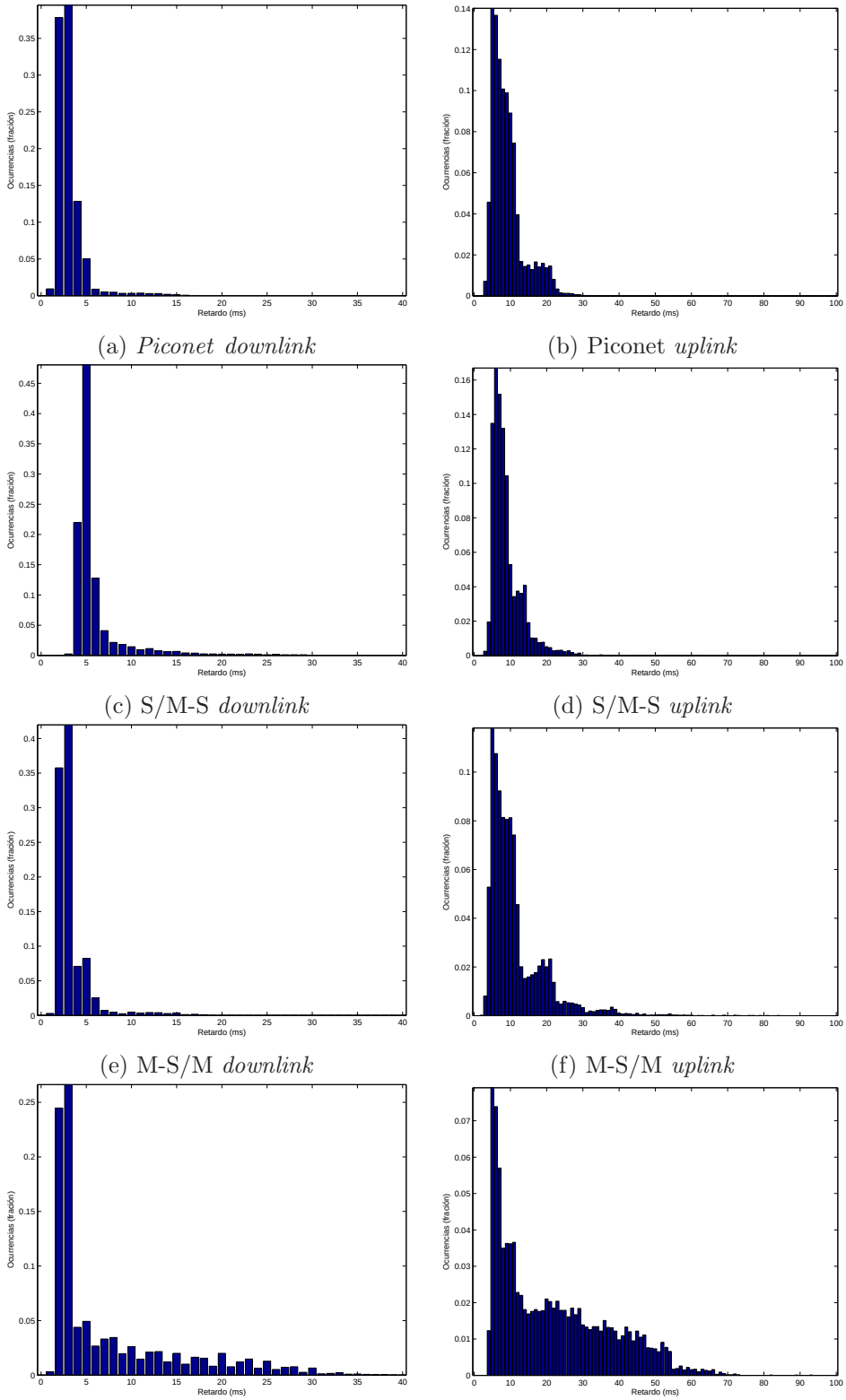
Figura 2.46: Variación del retardo medio en sentido $M \leftrightarrow S/S$ en función del parámetro *latency* (T_{poll})

Parámetro (ms)/Escenario		$M \leftrightarrow S$	$S/M \leftrightarrow S$	$M \leftrightarrow S/M$	$M \leftrightarrow SS$
Downlink	Media	3,53	6,54	3,73	8,10
	Mediana	3,1	5,5	3,2	3,8
Uplink	Media	9,57	9,37	11,77	22,47
	Mediana	8,5	8,1	9,4	19,0

Tabla 2.3: Estadísticas tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 10 y 30 ms

Las medidas presentadas en este mismo escenario han sido repetidas para distintas tasas de llegada de paquetes. Los resultados obtenidos son similares para menores tasas de llegada (mayor tiempo entre paquetes), ya que las condiciones en las que se ha realizado la prueba determinan que no se acumulan paquetes en las colas de transmisión al ser el retardo obtenido en la mayor parte de los paquetes inferior al tiempo mínimo entre llegadas. Los resultados para un tiempo entre llegadas menor, se recogen en la figura 2.47 y en la tabla 2.3. En este caso, sí que se acumulan paquetes en las colas de transmisión, lo que redunda en un descenso del retardo medio en sentido *uplink* en todos los escenarios, debido a que cuando el maestro sondea al esclavo y comprueba que tiene más paquetes que servir, lo vuelve a sondear sin esperar a que pase un tiempo T_{Poll}. En sentido *downlink*, en el que los paquetes son enviados por el maestro casi en cuanto están disponibles, los retardos se mantienen en niveles similares (*piconet* y *bridge M/S* o aumentan ligeramente (*bridge S/S*), según el escenario.

Se han realizado también pruebas con un tamaño de paquete mayor, en cuyo caso los envíos se fragmentan en varios paquetes. Los resultados se muestran en la figura 2.48 y en la tabla 2.4.



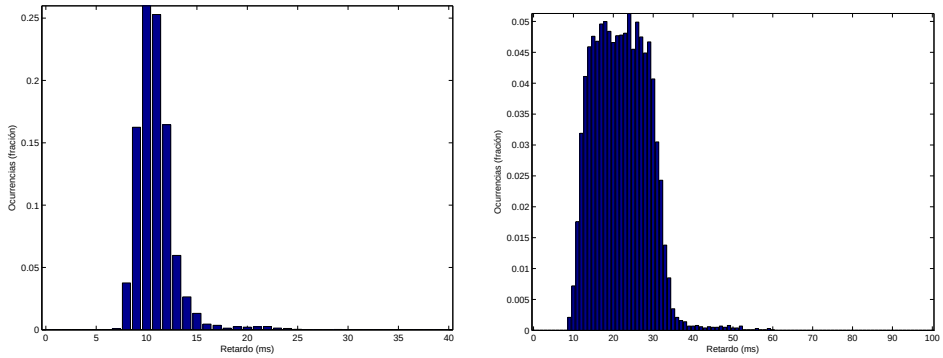
(a) *Piconet downlink* (b) Piconet *uplink*

(c) S/M-S *downlink* (d) S/M-S *uplink*

(e) M-S/M *downlink* (f) M-S/M *uplink*

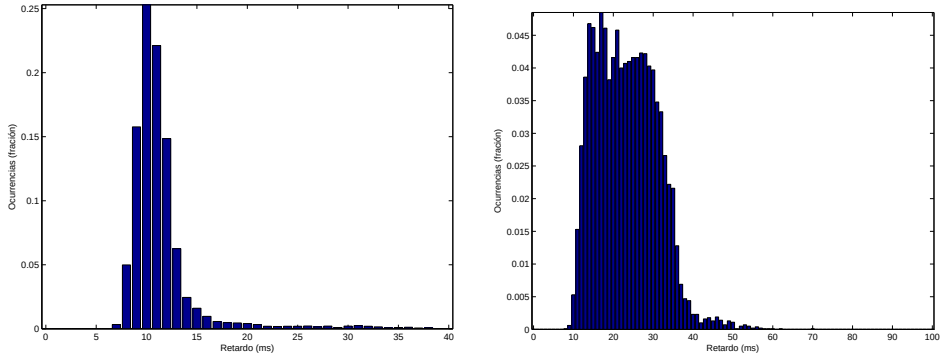
(g) M-S/S *downlink* (h) M-SS *uplink*

Figura 2.47: Resultados tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 10 y 30 ms



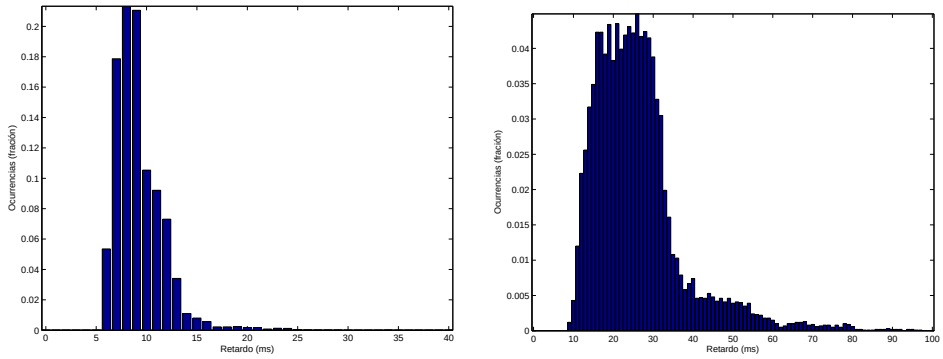
(a) *Piconet downlink*

(b) *Piconet uplink*



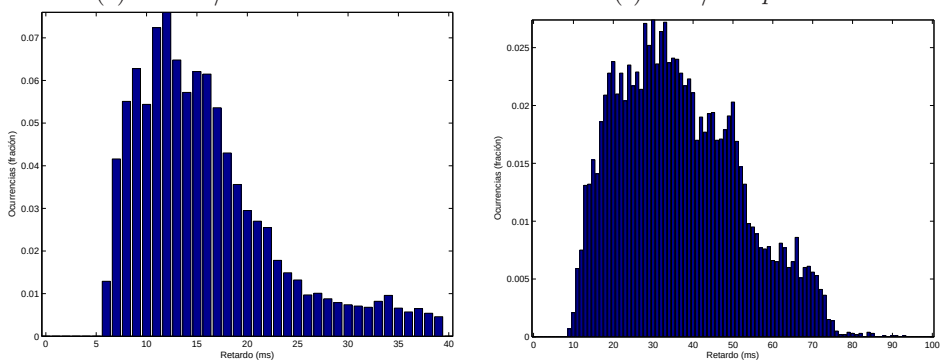
(c) $S/M - S$ *downlink*

(d) S/M-S *uplink*



(e) $M - S/M$ *downlink*

(f) M-S/M *uplink*



(g) $M - S/S$ *downlink*

(h) M-S/S *uplink*

Figura 2.48: Resultados tráfico asimétrico paquetes de 1000 bytes enviados con una separación entre 80 y 120 ms

Parámetro (ms)/Escenario		$M \leftrightarrow S$	$S/M \leftrightarrow S$	$M \leftrightarrow S/M$	$M \leftrightarrow SS$
Downlink	Media	11,40	12,03	9,72	16,83
	Mediana	11,14	11,16	9,25	15,04
Uplink	Media	22,56	23,88	26,86	37,18
	Mediana	22,35	23,43	24,84	35,39

Tabla 2.4: Estadísticas tráfico asimétrico paquetes de 1000 bytes enviados con una separación entre 80 y 120 ms

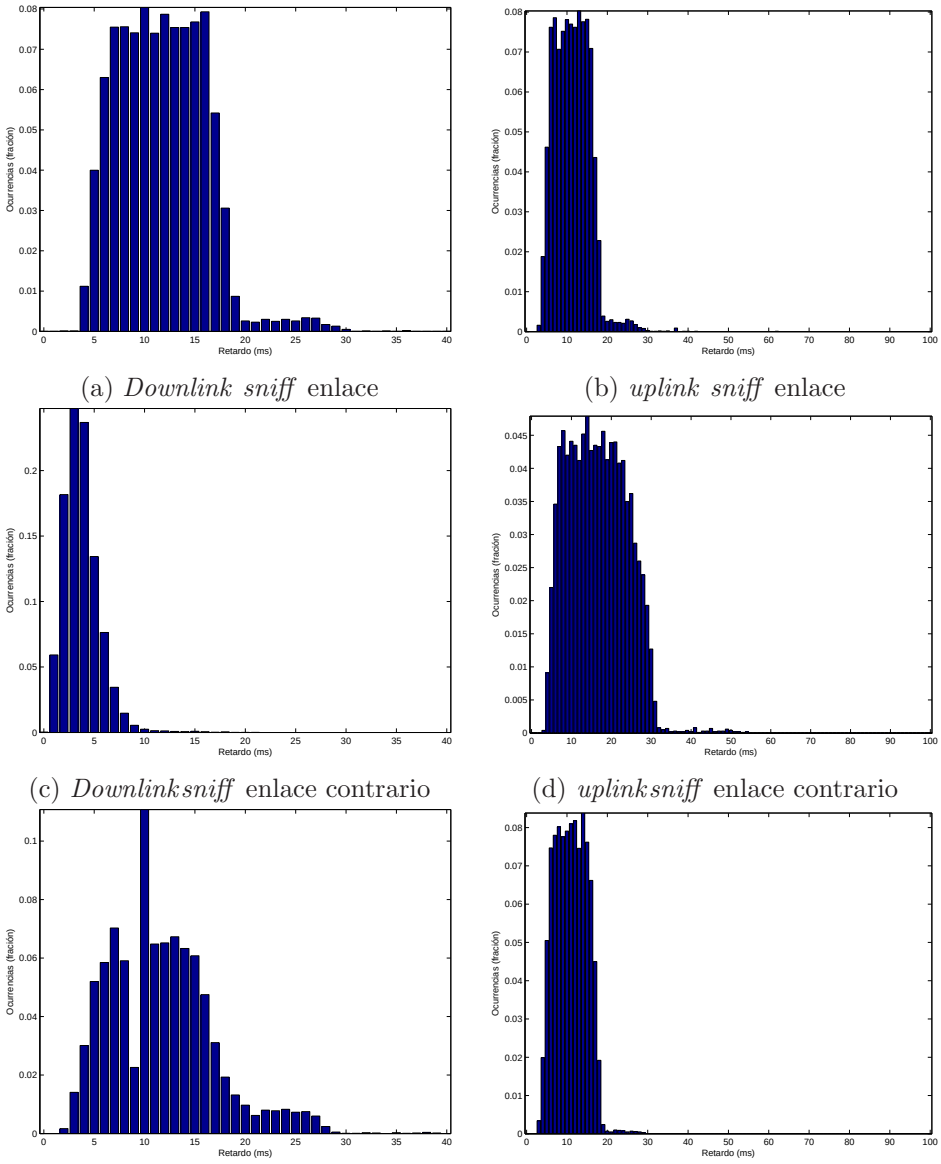
2.5.4. Utilización del modo *sniff*

En los resultados mostrados hasta ahora se pone de manifiesto que en un dispositivo comercial que participa como *bridge* en más de una *piconet* no existe realmente ninguna coordinación que permita acordar de antemano en qué *slots* va a estar presente el dispositivo en cada *piconet*. Un mecanismo que posibilita a los dispositivos esclavos la negociación de una ventana temporal en la que ser direccionados por el maestro y un periodo de inactividad en el que el maestro queda libre de sondear a los esclavos, es el modo *sniff*, que además permite a los dispositivos ahorrar energía. Cabe pues plantearse si a través de la configuración del modo *sniff* es posible mejorar el rendimiento de los dispositivos *bridge* en una *scattternet*. En esta sección se presentan los resultados de aplicar dicho modo en el escenario básico presentado anteriormente.

En general, las diferentes pruebas realizadas, de las cuales se presentarán sólo algunos resultados, indican que el modo *sniff* permite mejorar el comportamiento del sentido de transmisión *uplink* en las *scatternets* con *bridge* ($M \leftarrow S/S$), a costa de penalizar algo el sentido de comunicación *downlink* ($M \rightarrow S/S$). A nivel HCI, el modo *sniff* se configura mediante 4 parámetros N_{sniff_min}, N_{sniff_max}, $N_{attempt}$ y $N_{timeout}$. Por sencillez, en las pruebas que se presentan aquí, se ha configurado N_{sniff_max}=N_{sniff_min}, de forma que el periodo de *sniff* se fija al valor indicado y $N_{attempt} = N_{timeout} = 1$, de manera que el nodo esclavo permanece un único *slot* $M \rightarrow S$ a la espera de que le transmitan algún paquete, esperando un *slot* $M \rightarrow S$ adicional cada vez que se realice una comunicación. El *sniff* se configura independientemente para cada enlace, por lo que en cada escenario es posible poner en modo *sniff* uno de los enlaces o ambos a la vez, lo cual se ha tenido en cuenta para la realización de las pruebas.

Las figuras 2.49, 2.50 y 2.51 muestran los resultados para las diferentes topologías cuando el propio enlace medido está bajo el modo *sniff* (subfiguras a y b), cuando el enlace contrario es el que está configurado en modo *sniff* (subfiguras c y d) y con ambos enlaces configurados en modo *sniff* (subfiguras e y f). Los estadísticos (media y mediana) se recogen en la tabla 2.5. Todos estos resultados corresponden a un valor N_{sniff} de 20 *slots* (12,5 ms), para paquetes de 20 bytes enviados con un intervalo comprendido entre 80 y 120ms.

Los resultados parecen indicar que efectivamente, cuando se aplica el modo *sniff*, tanto en el propio enlace como en el contrario o en ambos, se obtiene una mejora del retardo *uplink* en la topología *S/S*, a costa de empeorar el retardo en sentido *downlink* si el propio enlace (o ambos) se configuran en modo *sniff*. En el caso de la topología $M \leftrightarrow S/M$, el



(a) *Downlink sniff* enlace

(b) *uplink sniff* enlace

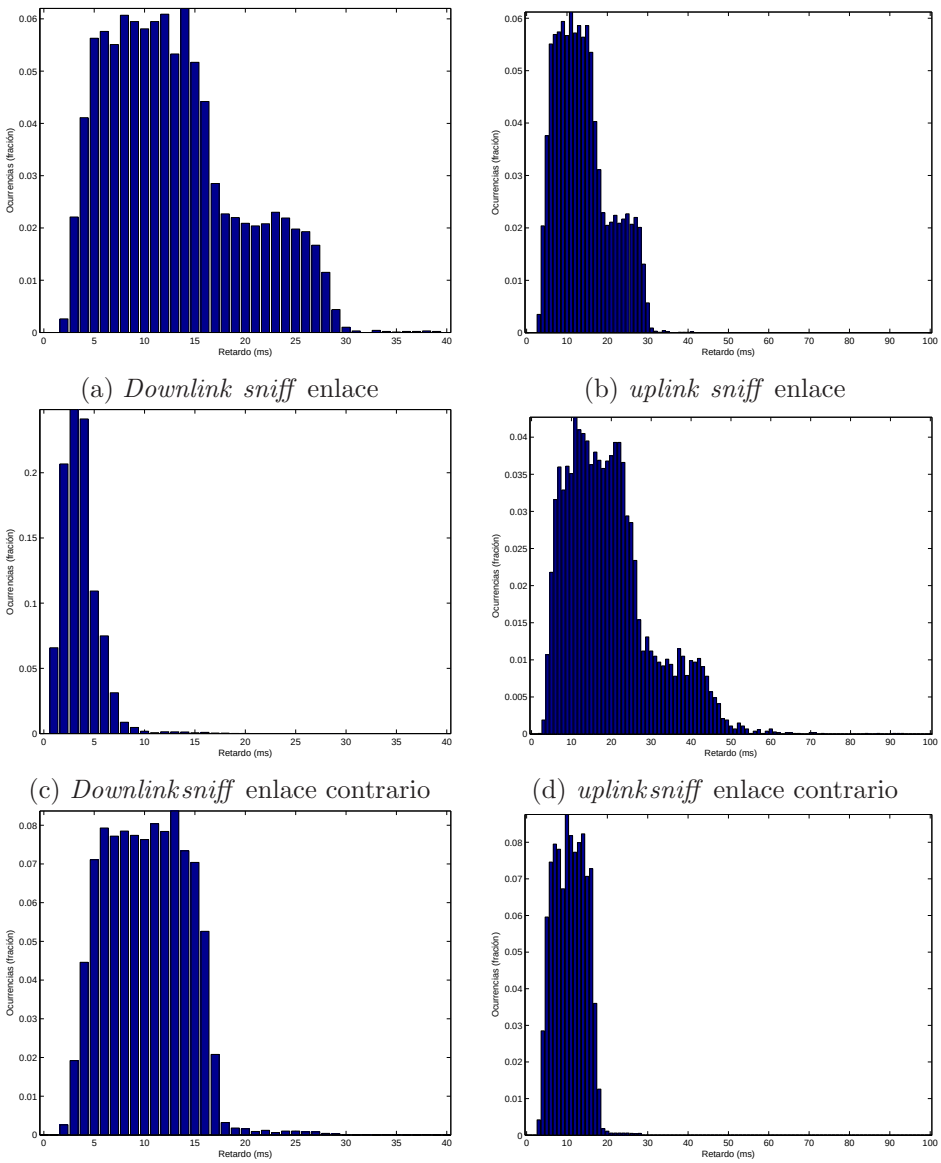
(c) *Downlinksniff* enlace contrario

(d) *uplinksniff* enlace contrario

(e) *Downlink sniff* Ambos

(f) *uplink sniff* Ambos

Figura 2.49: Retardo de los paquetes en el enlace $SM \leftrightarrow S$ para $N_{sniff} = 20$ (paquetes de 20 bytes con tiempo de envío entre 80 y 120ms)



(a) *Downlink sniff* enlace

(b) *uplink sniff* enlace

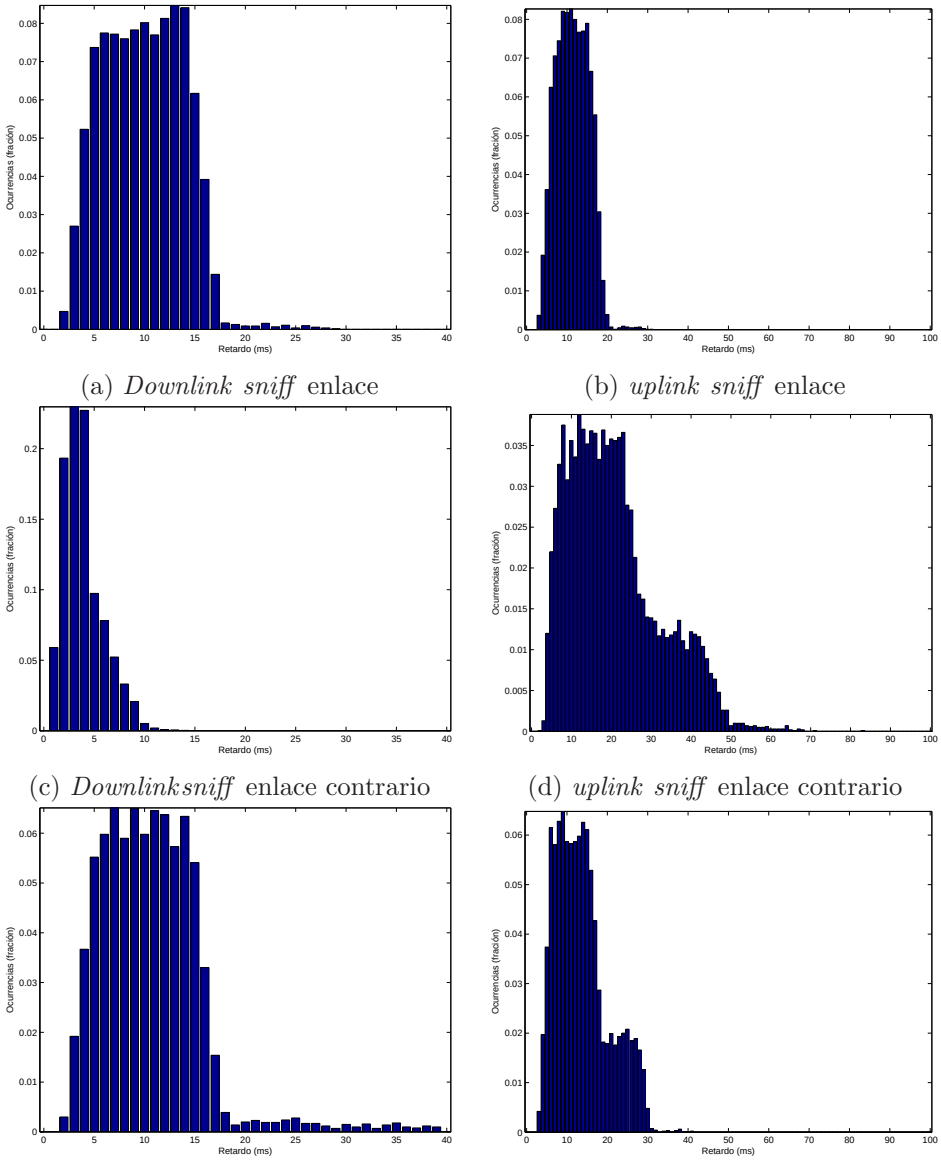
(c) *Downlinksniff* enlace contrario

(d) *uplinksniff* enlace contrario

(e) *Downlink sniff* Ambos

(f) *uplink sniff* Ambos

Figura 2.50: Retardo de los paquetes en el enlace $M \leftrightarrow SM$ para $N_{sniff} = 20$ (paquetes de 20 bytes con tiempo de envío entre 80 y 120ms)



(a) *Downlink sniff* enlace

(b) *uplink sniff* enlace

(c) *Downlinksniff* enlace contrario

(d) *uplink sniff* enlace contrario

(e) *Downlink sniff* Ambos

(f) *uplink sniff* Ambos

Figura 2.51: Retardo de los paquetes en el enlace $M \leftrightarrow SS$ para $N_{sniff} = 20$ (paquetes de 20 bytes con tiempo de envío entre 80 y 120ms

Parámetro (ms)/Escenario			$S/M \leftrightarrow S$	$M \leftrightarrow S/M$	$M \leftrightarrow SS$
Sniff enlace	*Downlink*	Media	12,27	13,47	10,43
		Mediana	12,09	12,44	10,40
	Uplink	Media	11,89	14,64	11,92
		Mediana	11,69	13,59	11,84
Sniff complementario	*Downlink*	Media	4,27	4,13	4,42
		Mediana	4,04	3,92	4,07
	Uplink	Media	17,19	20,36	21,30
		Mediana	16,86	18,54	19,37
Sniff ambos	*Downlink*	Media	50,22	10,71	32,64
		Mediana	11,69	10,66	12,16
	Uplink	Media	11,50	11,28	14,26
		Mediana	11,46	11,24	13,28

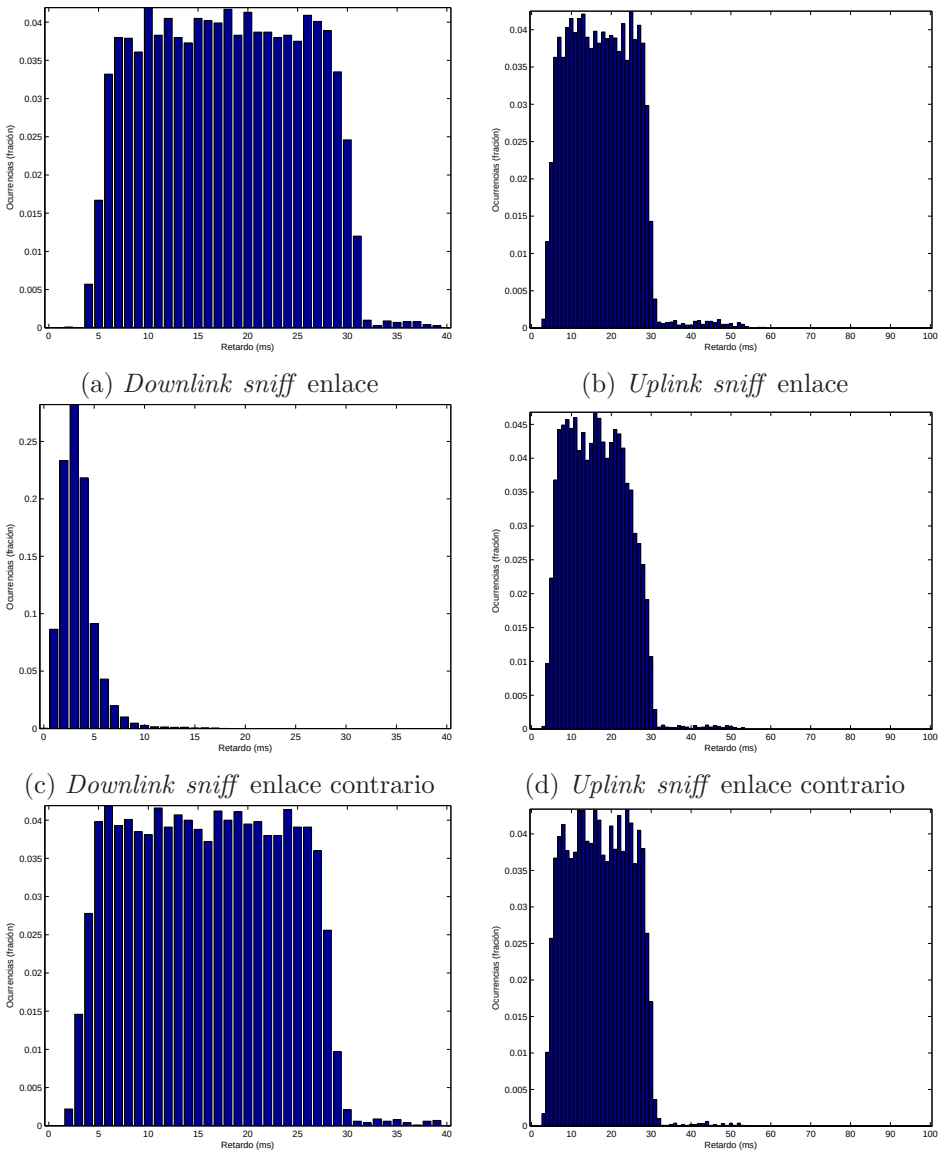
Tabla 2.5: Estadísticas para $N_{sniff} = 20$ (tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 80 y 120 ms)

modo *sniff* no parece impedir el hecho de que de vez en cuando el *bridge* se pierda el sondeo por parte de su maestro, excepto cuando se configura en ambos enlaces, en cuyo caso sí parece tener un claro beneficio en el sentido *uplink*. En este caso hay que tener en cuenta que el el valor de $N_{attempt}$ se ha puesto inferior al valor de T_{poll}, por lo que su activación mejora también el rendimiento del enlace ascendente en una *piconet*, lo que parece indicar que al negociar el modo *sniff* el maestro realiza el sondeo del esclavo en cada periodo de *sniff* aunque no tenga por qué. En este sentido, al realizar las pruebas se ha encontrado que valores muy agresivos de N_{sniff} (periodos muy reducidos) vuelven la red inestable, sobre todo cuando se intenta aplicar en todos los enlaces.

A continuación se incluye (figuras 2.52, 2.53, 2.54 y tabla 2.6) la misma batería de pruebas realizadas con $N_{sniff} = 40$ *slots* (25ms), que justo coincide con el tiempo T_{poll} configurado por defecto. En este caso se observa de nuevo la mejora que supone el establecer el modo *sniff* en ambos enlaces en el retardo del sentido *uplink*, especialmente en el bridge *S/S*, a costa de empeorar el comportamiento en sentido *downlink*. En estas pruebas ocurre un fenómeno curioso en el escenario $M \leftrightarrow S/M$, cuando dicho enlace se encuentra en modo *sniff* y el enlace complementario $S/M \leftrightarrow S$ no está en modo *sniff*. En este escenario algunos paquetes tienen un retardo elevado, lo cual empeora bastante el valor medio del retardo. Este resultado, que se ha comprobado repitiendo varias veces la prueba con otros módulos, se explicaría si el mecanismo que determina el sondeo que realiza el *bridge S/M* a su esclavo se realiza en cualquier momento en lugar de hacerlo en el tiempo de inactividad del que dispone al haber negociado un periodo de *sniff* con su maestro. Cuando se configuran ambos enlaces del *bridge S/M* en modo *sniff* este problema ya no aparece.

2.5.5. Configuraciones híbridas

Hasta ahora se han estudiado los escenarios más simples de un *bridge* en una *scatternet*: el *bridge S/M* con un sólo esclavo y el *bridge S/S* puro (sin esclavos). En esta



(a) *Downlink sniff* enlace

(b) *Uplink sniff* enlace

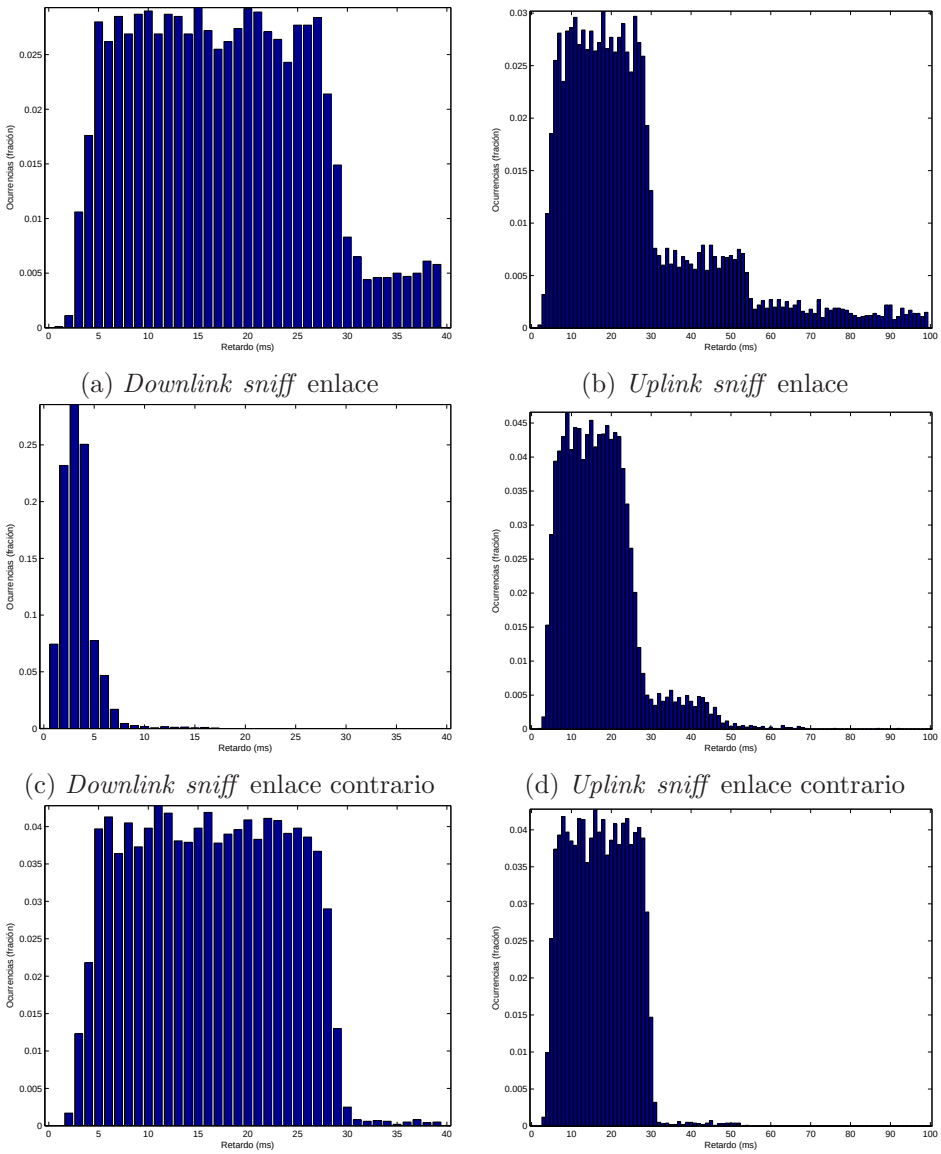
(c) *Downlink sniff* enlace contrario

(d) *Uplink sniff* enlace contrario

(e) *Downlink sniff* Ambos

(f) *Uplink sniff* Ambos

Figura 2.52: Retardo de los paquetes en el enlace $SM \leftrightarrow S$ para $N_{sniff} = 40$ (paquetes de 20 bytes con tiempo de envío entre 80 y 120ms)



(a) *Downlink sniff* enlace

(b) *Uplink sniff* enlace

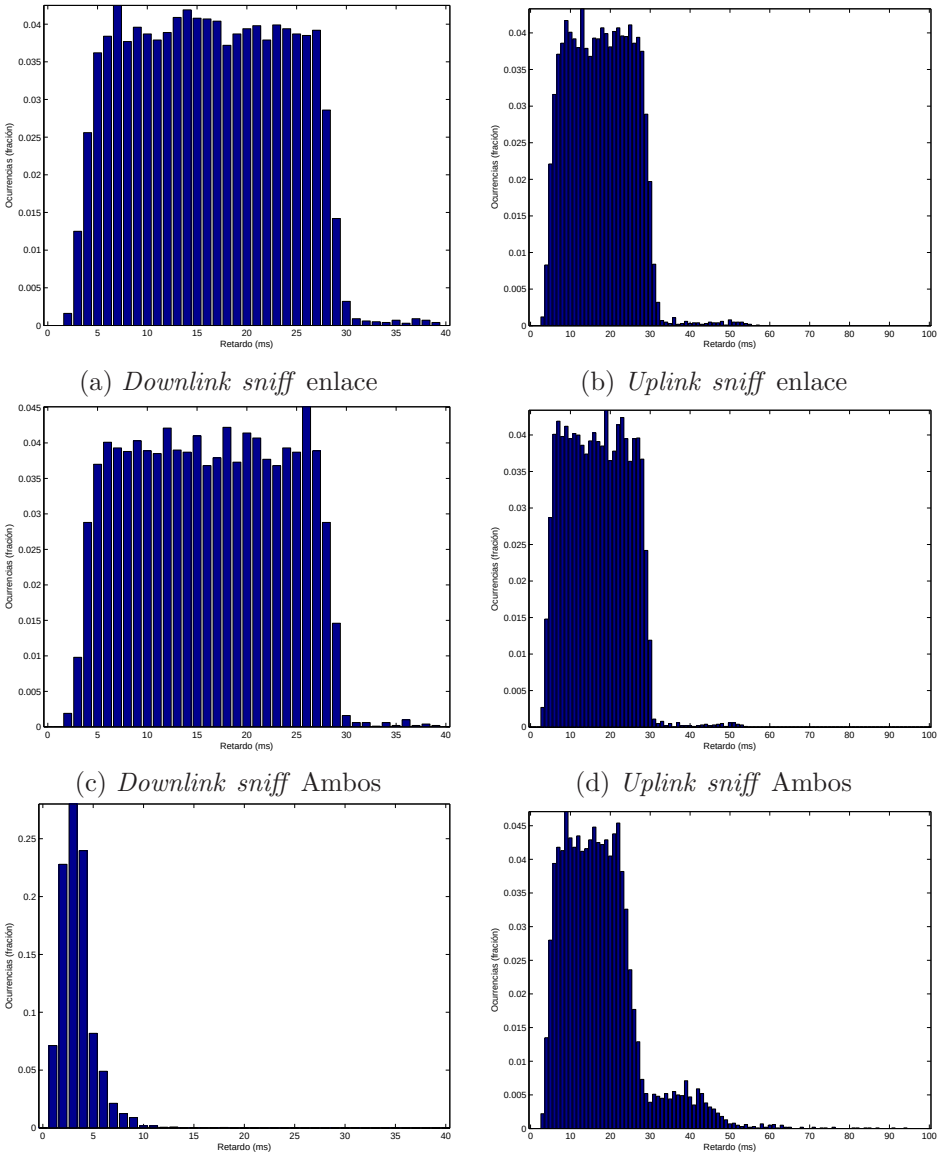
(c) *Downlink sniff* enlace contrario

(d) *Uplink sniff* enlace contrario

(e) *Downlink sniff* Ambos

(f) *Uplink sniff* Ambos

Figura 2.53: Retardo de los paquetes en el enlace $M \leftrightarrow S/M$ para $N_{sniff} = 40$ (paquetes de 20 bytes con tiempo de envío entre 80 y 120ms)



(a) *Downlink sniff* enlace

(b) *Uplink sniff* enlace

(c) *Downlink sniff* Ambos

(d) *Uplink sniff* Ambos

(e) *Downlinksniff* enlace contrario

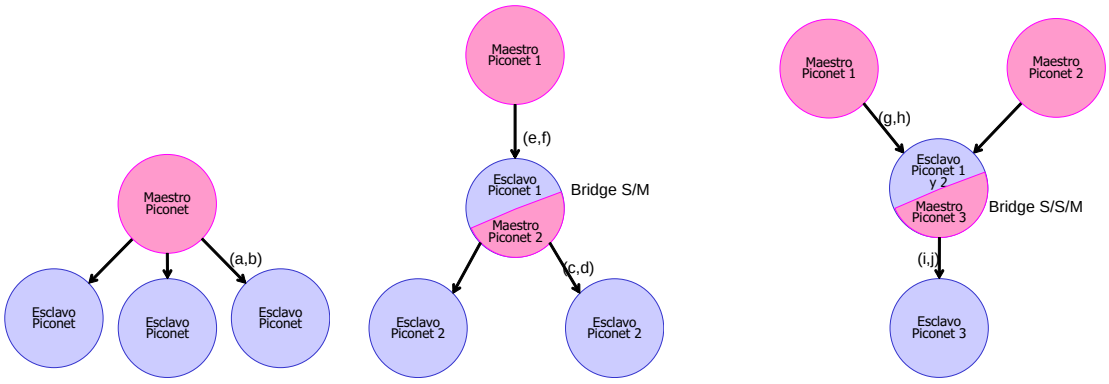
(f) *Uplinksniff* enlace contrario

Figura 2.54: Retardo de los paquetes en el enlace $M \leftrightarrow S/S$ para $N_{sniff} = 40$ (paquetes de 20 bytes con tiempo de envío entre 80 y 120ms

Parámetro (ms)/Escenario			$S/M \leftrightarrow S$	$M \leftrightarrow S/M$	$M \leftrightarrow S/S$
Sniff enlace	*Downlink*	Media	18,56	38,51	16,86
		Mediana	18,36	21,96	16,62
	Uplink	Media	18,12	34,78	18,24
		Mediana	17,84	23,06	18,14
Sniff complementario	*Downlink*	Media	3,88	3,86	3,97
		Mediana	3,65	3,68	3,72
	Uplink	Media	17,09	17,76	18,05
		Mediana	16,84	16,64	16,70
Sniff ambos	*Downlink*	Media	16,58	16,85	16,80
		Mediana	16,50	16,68	16,72
	Uplink	Media	17,78	17,85	17,50
		Mediana	17,62	17,73	17,38

Tabla 2.6: Estadísticas para $N_{sniff} = 40$ (tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 80 y 120 ms)

sección se incluyen algunos resultados obtenidos en algunos escenarios más complejos, que se muestran en la figura 2.55. En el escenario II se añade un segundo esclavo al *bridge S/M*, mientras que en el escenario III se crea un *bridge* híbrido que tiene dos maestros y un hijo esclavo, por lo que participa en 3 *piconets* diferentes. El escenario (I) corresponde a una *piconet* con tres dispositivos esclavos, y se incluye como referencia. Como ya se ha mencionado previamente, los módulos *Bluetooth* utilizados no funcionan de forma suficientemente estable en un escenario con tres dispositivos padre, por lo que dicha configuración no se ha evaluado. En cada escenario se han caracterizado los enlaces que se marcan en la figura 2.55, tanto en sentido *uplink* como *downlink*.



I) Piconet tres esclavos

II) *Scatternet* con *bridge S/M* con dos esclavos

III) *Scatternet* con *bridge S/S/M*

Figura 2.55: Configuraciones híbridas evaluadas

En general los resultados obtenidos en este escenario se aproximan bastante a lo que cabría esperar a raiz del estudio realizado en las secciones anteriores. Los histogramas del retardo para paquetes de 20 bytes enviados con una distribución uniforme entre 80 y 120ms, se muestran por ejemplo en la figura 2.56 y tabla 2.7 para los enlaces marcados en la figura 2.55. En ellas puede observarse que el comportamiento del enlace $SSM \leftrightarrow S$

Parámetro (ms)/Escenario		$M \leftrightarrow S$	$S/M \leftrightarrow S$	$M \leftrightarrow S/M$	$M \leftrightarrow SSM$	$SSM \leftrightarrow S$
Downlink	Media	3,76	6,97	4,02	8,58	8,87
	Mediana	3,30	5,80	3,30	5,80	6,50
Uplink	Media	15,69	16,48	22,35	31,93	16,12
	Mediana	15,50	16,20	18,60	27,90	15,70

Tabla 2.7: Estadísticas tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 80 y 120 ms

(subfiguras i y j) es bastante similar al comportamiento $SM \leftrightarrow S$ (subfiguras c y d), ya que en sentido *uplink* el *bridge* sondea de vez en cuando a su esclavo, y en sentido *downlink* procede a transmitir el paquete en cuanto este se recibe, abandonando en ambos casos las *piconets* de sus padres. Esto hace que el enlace $M \leftrightarrow SSM$ se comporte algo peor que en el caso del *bridge S/S* puro, ya que ahora además de repartirse entre la *piconet* de ambos padres, tiene que dedicar cierta cantidad de tiempo a atender su propia *piconet*. También resulta algo peor el comportamiento en el enlace $M \leftrightarrow S/M$, ya que el bridge se tiene que encargar de otro dispositivo esclavo, aumentando por tanto la probabilidad de no estar presente cuando su dispositivo *padre maestro* le sondee.

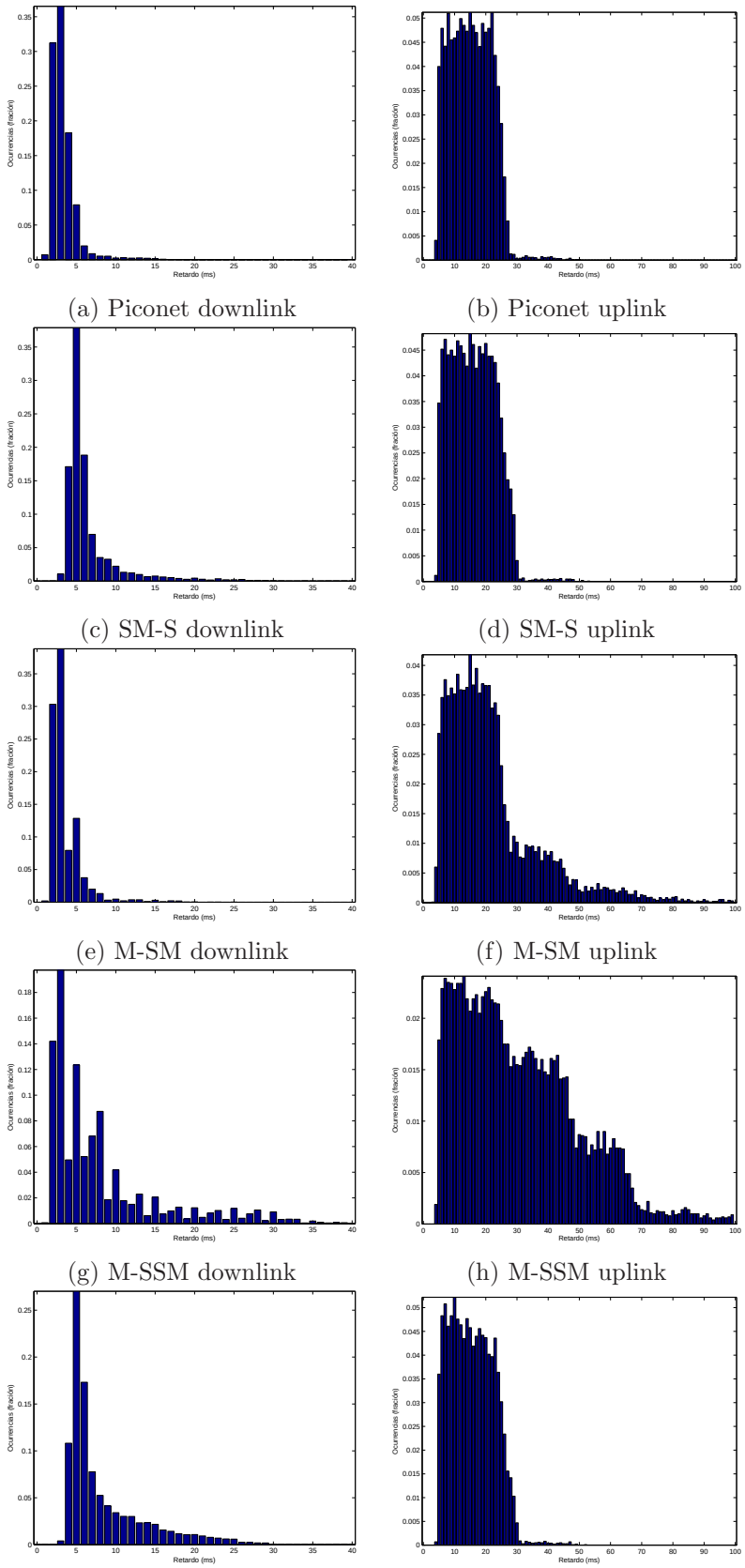
2.6. Conclusiones

La tecnología *Bluetooth* es desde hace algunos años uno de los interfaces de comunicación presente en la mayoría de terminales móviles y dispositivos de comunicación personales, por lo que es una opción competitiva para que objetos inteligentes puedan interaccionar con dichos terminales mediante la formación de redes expontáneas, incluso aunque la tecnología no fuese inicialmente concebida para ello.

Bluetooth presenta dos limitaciones importantes a la hora de servir como tecnología de red de área personal para éstos dispositivos, que son un consumo que puede considerarse elevado para dispositivos que deben permanecer alimentados a batería durante periodos de tiempo largos, y el pequeño número de dispositivos que pueden interconectarse con la topología básica en estrella que impone la *piconet*. Para aliviar en parte estas limitaciones, la norma establece diversos modos de funcionamiento que permiten reducir el consumo, así como la posibilidad de formar redes *scatternet* mediante dispositivos puente que participan en más de una *piconet*.

En este capítulo se ha realizado en primer lugar la caracterización del comportamiento energético de diversas implementaciones de dispositivos comerciales en diferentes modos de funcionamiento para las topologías *piconet* y *scatternet*, proponiendo un modelo del comportamiento energético de los dispositivos cuando participan en más de una piconet para extender el área de cobertura de la red.

El funcionamiento de los dispositivos puente que sirven de enlace entre diferentes *piconets* al formar una *scatternet* no está apenas definido por el estándar, por lo que se han realizado pruebas sobre dispositivos reales para comprobar cómo se comportan, comprobando que en la práctica el comportamiento presenta bastante más limitaciones y peor



(a) Piconet downlink (b) Piconet uplink

(c) SM-S downlink (d) SM-S uplink

(e) M-SM downlink (f) M-SM uplink

(g) M-SSM downlink (h) M-SSM uplink

(i) SSM-S downlink (j) SSM-S uplink

Figura 2.56: Estadísticas tráfico asimétrico paquetes de 20 bytes enviados con una separación entre 80 y 120 ms

comportamiento que el postulado por algunos trabajos de investigación. Finalmente se ha comprobado que parte de los problemas pueden ser atenuados utilizando los modos de bajo consumo, permitiendo además reducir el gasto de batería realizado por los dispositivos.

Capítulo 3

Evaluación empírica y modelado de WPAN basadas en *ZigBee*

3.1. Introducción

En este capítulo se recogen las aportaciones realizadas en relación con la evaluación de redes de área personal basadas en tecnología *ZigBee/802.15.4*. En la sección 3.2 se detallan los fundamentos del funcionamiento de esta tecnología, derivados del estudio de las diferentes versiones del estándar. Al igual que en el capítulo anterior, se realiza una revisión completa de todos los niveles de la arquitectura de protocolos, aunque los aspectos más relevantes para el estudio realizado en este capítulo se concentran principalmente en la sección 3.2.4.4 en la que se describe el funcionamiento del mecanismo de control de acceso al medio.

Manteniendo la estructura seguida en el capítulo anterior, en la sección 3.3 se realiza una revisión de los diferentes campos de estudio en los que se centran los diferentes trabajos y propuestas realizados por la comunidad investigadora. Por último, los estudios realizados dentro del marco de la presente tesis se presentan en la secciones 3.4 y 3.5.

3.2. Fundamentos de *ZigBee* e IEEE 802.15.4

ZigBee [Zbsb] es un estándar que define un conjunto de protocolos para redes inalámbricas de área personal en bandas sin licencia, y está principalmente enfocado a sistemas que no requieran una tasa de datos muy alta pero que impliquen conectar un elevado número de dispositivos, algunos de ellos alimentados con batería. Entre sus aplicaciones principales destacan la domótica, y las redes de sensores y actuadores. La principal ventaja que presenta frente a otras tecnologías competidoras como *Bluetooth Low Energy* es que el estándar sí que contempla la formación de redes con un gran número de dispositivos y con encaminamiento multisalto, de forma que puede cubrirse un área más extensa mediante dispositivos de alcance radio limitado.

Existe tradicionalmente cierta confusión entre *ZigBee* y el estándar IEEE 802.15.4 [Ieea], que aunque están relacionados no son términos equivalentes. El estándar *ZigBee* ha sido desarrollado y es mantenido por la *ZigBee Alliance*, que agrupa a diversas compañías

de desarrollo software y de la industria de los semiconductores. Esta organización ha adoptado el estándar 802.15.4 del IEEE como implementación de los niveles inferiores de la pila de protocolo (capa física y control de acceso al medio). El resto de protocolos de nivel superior, necesarios para dar soporte a funciones como el enrutamiento o la búsqueda de servicios, sí son específicamente definidos y mantenidos por la propia organización. La norma IEEE 802.15.4 no especifica el comportamiento de los protocolos de nivel superior al MAC, sino que por el contrario, está concebida para dar soporte a distintos estándares que definan estos niveles, como pueden ser RF4CE [Rf4] y 6LowPan [SB11], además del propio *ZigBee*, por lo que su diseño es lo suficientemente flexible como para poder cubrir los distintos niveles de complejidad requeridos por los diferentes escenarios en los que se utilice. Por otra parte, los dispositivos *ZigBee* no implementan o no utilizan todas las formas de comunicación previstas en el estándar IEEE 802.15.4, sino sólo un subconjunto de ellas. A pesar de esta separación, ambos estándares comparten una serie de conceptos y terminologías comunes, por lo cual su funcionamiento se describe de forma conjunta en esta sección, intentando siempre dejar claras las diferencias y coincidencias entre ambos.

La primera versión de ambos estándares es del año 2003, si bien desde entonces sus especificaciones han sufrido varias revisiones en las que se han ido introduciendo características adicionales y algunas correcciones. En particular, el estándar de *ZigBee* sufrió una profunda revisión en los años 2006 y 2007, dando lugar a la primera versión que se implementó de forma masiva en productos funcionales. Posteriormente, en el año 2012 se realizó una nueva revisión [Zbsa], en la que se incorporan algunas características adicionales y se establece una separación entre las especificaciones del núcleo de los protocolos (*Core Specification*), que permite la interoperabilidad de los dispositivos a nivel de comunicación, y la especificación de perfiles de aplicación (*Profile Specification*), que garantiza su correcta interacción como componentes de una aplicación concreta (Salud, Hogar Inteligente, *Smartgrid*, etc.). En el caso de IEEE 802.15.4, se publicó una revisión importante del mismo en 2006 [Ieeb], y posteriormente el grupo de trabajo que lo mantiene ha ido añadido anexos adicionales, principalmente destinados a especificar nuevas variantes de la capa física adaptadas a las regulaciones y bandas de frecuencia libres de diferentes países, que fueron posteriormente incorporadas en un nueva versión unificada del estándar liberada en el año 2011 [Ieec]. La última revisión del estándar se ha realizado en 2015, y además de nuevas propuestas de implementación para la capa física en mayor número de bandas recoge una nueva especificación para la capa MAC, denominada 802.15.4e, que persigue reducir el consumo y mejorar el uso del ancho de banda mediante la utilización de un esquema de temporización basado en TDMA (denominado TSCH) [Ieed].

3.2.1. Arquitectura y topología de la red

Para garantizar la suficiente flexibilidad, IEEE 802.15.4 define diversas topologías de red en las que pueden participar dispositivos de distinto tipo y que realizan diferentes funciones. Atendiendo a sus capacidades de comunicación el estándar define dos tipos diferentes de dispositivo, los que denomina dispositivos de funcionalidad completa o FFD (*Full Function Device*) y los que denomina de funcionalidad reducida o RFD (*Reduced Function Device*). Los dispositivos de funcionalidad completa (FFD) soportan todas las

posibles capacidades de comunicación, pudiendo comunicarse con cualquier dispositivo en la red, y son capaces de asumir cualquiera de los diferentes papeles (*roles*) que pueden desempeñar los dispositivos en la red. Por su parte, los dispositivos de funcionalidad reducida (RFD) poseen unas capacidades de comunicación más limitadas, ya que sólo pueden comunicarse con un dispositivo FFD, no implementan todas las posibles funciones definidas por el estándar y no son capaces de asumir cualquier papel en la red PAN. Esta diferenciación permite integrar en la red dispositivos con menores exigencias de capacidad de proceso, memoria y alimentación.

3.2.1.1. Roles de los dispositivos

En una red IEEE 802.15.4 los dispositivos pueden desempeñar tres diferentes funciones o *roles*: coordinador, coordinador PAN y dispositivo final. Un coordinador es un dispositivo de funcionalidad completa que es capaz de re-encaminar mensajes (es decir, recibirlos y volverlos a transmitir hacia otro dispositivo). En las redes PAN definidas por IEEE 802.15.4, uno de los múltiples dispositivos coordinadores que pueden formar parte de la red PAN realizará las funciones de controlador principal de la red, recibiendo el nombre de coordinador PAN. Si un dispositivo únicamente recibe o transmite sus propios mensajes y no procesa los mensajes para los que no es fuente ni destino, entonces se trata de un dispositivo final. El papel de coordinador o coordinador PAN sólo puede ser desempeñado por los dispositivos FFD, mientras que el de dispositivo final puede ser interpretado tanto por un FFD como por un RFD. Evidentemente, los RFD sólo pueden actuar como dispositivos finales.

ZigBee también define diferentes tipos de nodos con distinta funcionalidad según el papel que desempeñan en la red, y puesto que utiliza 802.15.4 como implementación de los niveles inferiores, cada tipo de nodo definido en *ZigBee* se corresponde con uno de los tipos definidos por IEEE 802.15.4. Sin embargo, la nomenclatura utilizada por *ZigBee* es algo diferente, lo cual frecuentemente lleva a cierta confusión. Como se detallará más adelante en la sección 3.2.5.1, *ZigBee* define tres tipos de nodos: coordinador (*coordinator*), *router* y dispositivo final (*end device*), que se corresponden respectivamente con los dispositivos de tipo coordinador PAN (*PAN coordinator*), coordinador (*coordinator*) y dispositivo (*device*) de 802.15.4.

3.2.1.2. Topologías de red

Si bien la formación y gestión de la red PAN compete a los niveles superiores de la pila de protocolo, 802.15.4 permite la formación de dos topologías de red diferentes de forma que puedan cubrirse las necesidades de los diferentes protocolos de red soportados: topología en estrella y topología *peer-to-peer*. En la topología en estrella, cada dispositivo de la red puede comunicarse únicamente con el coordinador PAN. En la topología *peer-to-peer*, cualquier dispositivo FFD (coordinador o coordinador PAN) puede comunicarse con cualquier otro que esté a su alcance, y los dispositivos RFD pueden comunicarse con uno cualquiera de los dispositivos coordinadores a su alcance, tanto si es el coordinador PAN como si no. En una misma red, sólo uno de los dispositivos FFD tiene el papel

de coordinador PAN. El coordinador PAN es el dispositivo encargado de formar la red, determinando su identificador, y de asignar direcciones a los dispositivos que se conecten a ella.

La topología *peer-to-peer* ofrece soporte para el establecimiento de redes más complejas que podrían ser adhoc, auto-organizadas, y con capacidad de auto-regeneración (*Self-healing*). Si se introducen restricciones adicionales en la comunicación entre los dispositivos que la forman, una red *peer-to-peer* puede tomar distintas formas, dando lugar por ejemplo a redes malladas, en las que no hay restricciones, o a redes en árbol, en las que se establece una jerarquía de comunicación en la que el coordinador PAN es el dispositivo raiz.

En cualquier caso, la organización de la red y los mecanismos para conseguir el encaminamiento de información entre nodos sin conexión directa, no son definidos por 802.15.4, pues corresponden a niveles superiores de la pila de protocolos. En el caso del estándar *ZigBee*, los mecanismos de nivel de red implementados permiten una organización de la comunicación de la red en estrella, en árbol (mediante un mecanismo de enrutamiento jerárquico) y en malla (con un mecanismo de búsqueda y actualización de rutas entre dispositivos). Tanto en la topología en árbol como en estrella, sólo los dispositivos *routers* o coordinadores *ZigBee* pueden re-encaminar paquetes, mientras que los dispositivos finales sólo pueden comunicarse con un dispositivo padre (que debe ser un *router* o el coordinador). El coordinador *ZigBee*, que tiene la función de coordinador PAN 802.15.4, es el encargado de la formación del árbol jerárquico mediante la dirección que le asigna a los demás dispositivos conforme se incorporan a la red, siendo además el nodo más alto de la jerarquía (raíz). La topología en red mallada (con búsqueda de rutas entre nodos) puede permitir una comunicación más óptima entre los dispositivos utilizando un menor número de saltos que la topología en árbol, pero impone algunas limitaciones en la configuración del funcionamiento de 802.15.4 que se comentarán más adelante, y que básicamente se traducen en que cuando se utiliza esta topología, los nodos *routers* y el coordinador deben permanecer con la radio activa durante todo el tiempo y únicamente los nodos hoja pueden entrar en modo de bajo consumo.

3.2.2. Arquitectura de protocolos

La arquitectura de protocolos completa de *ZigBee*, incluyendo los niveles que corresponden a 802.15.4, se muestra en la figura 3.1. Las capas física (PHY) y de acceso al medio (MAC) definidas por IEEE 802.15.4 habilitan la comunicación básica entre dispositivos con conectividad directa entre ellos. *ZigBee* añade una capa de red (NWK) que permite el transporte de datos en la red a través de varios saltos y por tanto habilita la comunicación entre dispositivos que carezcan de visibilidad directa, encargándose también de buscar y actualizar posibles rutas entre los dispositivos. Además de la capa de red, el estándar también establece una serie de componentes de nivel de aplicación (APL) que facilitan el desarrollo de la misma, permitiendo que cada dispositivo pueda ofrecer diversos servicios y habilitando un mecanismo para el descubrimiento de los mismos.

En la figura 3.1 se distingue entre el plano de datos y el plano de control o gestión en las comunicaciones entre diferentes niveles. El punto de acceso al servicio (SAP) del plano de control permite al protocolo de nivel superior establecer configuraciones u ordenar

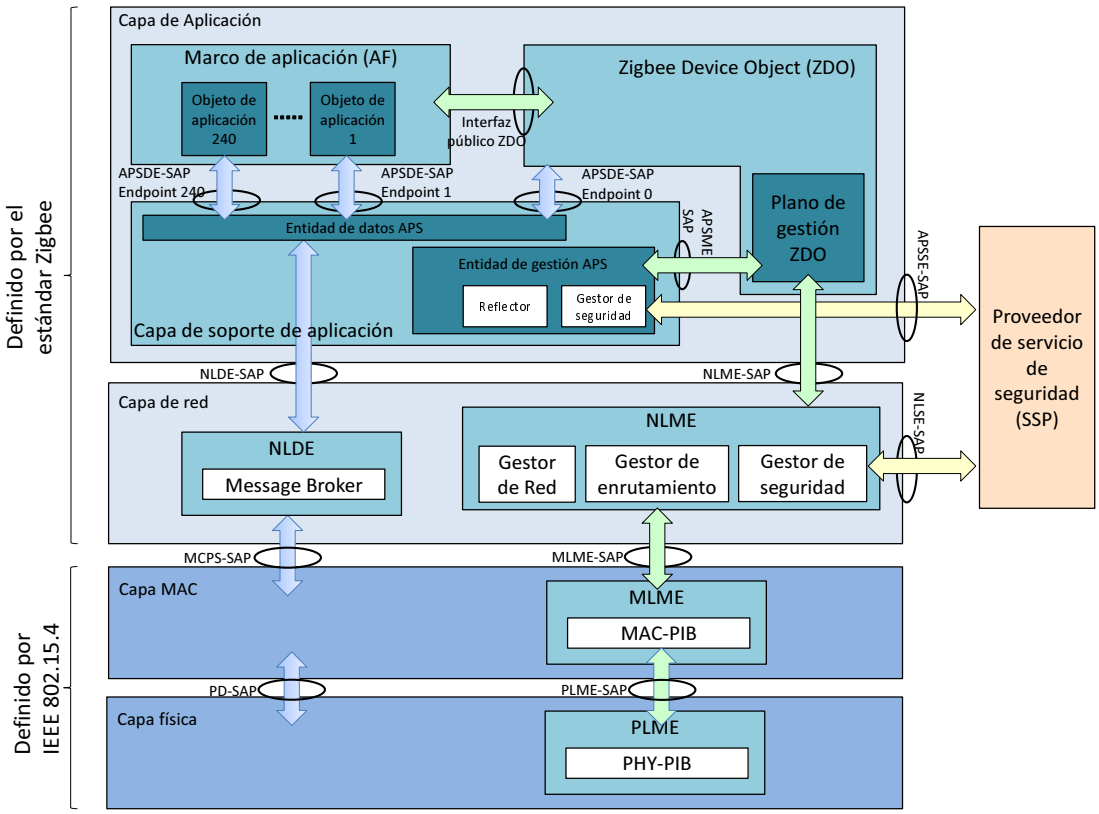


Figura 3.1: Arquitectura de protocolos *ZigBee* e IEE 802.15.4.

diferentes operaciones de gestión al protocolo de nivel inferior, mientras que punto de acceso al servicio de datos, permite controlar el envío y recepción de datos a través de los niveles inferiores.

El nivel de aplicación está basado en varios subniveles o componentes: La capa de soporte de aplicación (APS), el marco o *framework* de aplicación (AF) y el *ZigBee Device Object* (ZDO). La capa de soporte de aplicación hace de interfaz entre las aplicaciones y la capa de red, permitiendo la multiplexación del transporte de datos de las diversas aplicaciones a través de ésta. Para permitir esto, se asigna una dirección *endpoint* distinta a cada objeto de aplicación (de forma similar a lo que sería el concepto de puerto en los protocolos TCP/UDP). Los objetos de aplicación son definidos por los desarrolladores y fabricantes de productos y aplicaciones basadas en *ZigBee* y por tanto no son especificados por el *core* de la norma, aunque la *ZigBee Alliance* sí establece (en documentos separados) cómo deben implementarse determinadas aplicaciones o perfiles para garantizar la interoperabilidad entre fabricantes. Además, el *framework* de aplicación define una serie de componentes para facilitar la implementación de las mismas.

El *endpoint* 0 tiene un tratamiento especial, ya que se asigna para el envío por la red de datos correspondientes al *ZigBee Device Object*, que es un componente del nivel de aplicación definido por el estándar. Este componente engloba todas las funcionalidades que son comunes a todos los dispositivos que operan con tecnología *ZigBee* independientemente de su comportamiento a nivel de aplicación. Por ejemplo, es el encargado de determinar si el dispositivo se va a comportar como coordinador, *router*, o *end-device* y establecer los parámetros con los que van operar los niveles inferiores.

En las siguientes secciones se describe con algo más de detalle el funcionamiento de las diferentes capas.

3.2.3. Capa Física IEEE 802.15.4

La capa física gestiona la transmisión y recepción de datos usando determinados canales radio, además de permitir algunas operaciones relacionadas con los mismos como por ejemplo la detección del estado de ocupación del canal. La especificación de la capa física establece además qué bandas de frecuencia pueden ser utilizadas y la modulación utilizada para dividir cada banda en canales y codificar la información a transmitir.

Las especificaciones originales de 802.15.4 establecían tres posibles bandas de frecuencia en las que poder operar: 868 MHz, 915 MHz y 2,4 GHz, coincidiendo con las bandas de libre uso disponibles en Europa y Estados Unidos. En función del ancho de banda disponible, en cada banda se utiliza una técnica de modulación diferente (BPSK en las bandas por debajo de 1GHz y O-QPSK con ensanchado de espectro por secuencia directa en la de 2,45Ghz), dando lugar también a tasas de datos distintas. Como ya se ha comentado previamente, de acuerdo con la evolución que se ha ido produciendo en el campo de las tecnologías radio, el grupo de trabajo del 802.15.4 se ha encargado posteriormente de ir aumentando las técnicas de modulación y ensanchamiento de espectro disponibles para permitir operar en otras bandas (por ejemplo UWB para la banda de 3 a 10 Ghz) y/o mejorar la tasa de datos en una banda concreta.

De todas las alternativas disponibles, la mayor parte de implementaciones comerciales que han existido y existen en la actualidad utilizan la banda de 2,4 Ghz, que queda dividida en 16 canales espaciados 5 MHz (y numerados del 11 al 26), para los que se consigue una tasa de datos de 250 kbps mediante el empleo de una modulación O-QPSK con DSSS (ensanchado de espectro por secuencia directa). Según esta técnica, los bits a transmitir se agrupan y codifican en símbolos de 4 bits, y cada uno de los 16 posibles símbolos es codificado mediante una secuencia de 32 chips que serán los que finalmente se modulen. La tasa de símbolos para esta tecnología es de 62.5 ks/s y por tanto la duración de un símbolo (un parámetro importante en el funcionamiento del MAC, ya que mucho de los tiempos se especifican de forma relativa a este valor para independizar su especificación de la capa física) es de 16 μs

Independientemente de la tecnología utilizada para enviar información por el medio radio, entre las funciones que realiza la capa física en 802.15.4 se puede destacar:

- **Activar y desactivar el transceptor radio.** El transceptor radio tiene tres modos de funcionamiento: transmisión, recepción y sleeping (descanso). Bajo petición de la capa MAC la capa física debe conmutar entre estos tres estados. El estándar exige que la conmutación entre transmisión y recepción, o viceversa, se haga en menos de 12 símbolos (constante $aTurnaroundTime = 12$). En el caso mencionado de O-QPSK con DSSS en 2,4 GHz, este tiempo es de 192 μs.

- *Detección de energía en cada canal (ED,* Energy Detection *).* La capa física es la encargada de estimar el nivel de potencia que existe un determinado canal. Esta medida es utilizada por la capa de red como parte de su algoritmo para la elección

de canal, y también puede ser utilizada por los mecanismos de control de acceso al medio. El nivel de energía en la banda del canal de interés se promedia a lo largo de un periodo de al menos 8 símbolos (128 μs) para proporcionar la estimación, durante el cual la radio debe permanecer en modo recepción. Es importante destacar que la presencia de energía en un canal puede ser debida tanto a comunicaciones *ZigBee* en curso entre otros dispositivos como a otras comunicaciones que puedan coexistir en la misma banda.

- **Detección de portadora (CS, *carrier sense*).** Mediante esta característica, la capa física es capaz de estimar si hay en curso otra comunicación conforme con el estándar 802.15.4 en un determinado canal. La diferencia con el método anterior es que se busca específicamente la presencia de señales compatibles con el estándar independientemente del nivel de energía observado en el canal.

- **Evaluación de canal libre (CCA, *Clear Channel Assessment*).** Este indicador será utilizado por el algoritmo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) como se verá más adelante como indicador de que el canal está ocupado (o no) por otra comunicación. En base a los mecanismos anteriormente mencionados (detección de portadora y detección de energía), existen diversos modos de realizar el CCA que pueden configurarse: que la energía medida (ED) sea superior a un umbral, que se detecte una señal compatible con 802.15.4 (CS) o una combinación (*or* u *and*) de ambos. El periodo para medirlo es de 8 símbolos.

- **Indicación de la calidad de enlace (LQI, *Link Quality Indicator*).** Mide la calidad de los paquetes de datos recibidos por el dispositivo mediante diferentes parámetros: La potencia de señal recibida en dBm (RSS) y la relación señal ruido (SNR) estimada. La calidad del enlace es reportada a la capa MAC y queda disponible para su análisis los niveles superiores (red y aplicación) que los pueden utilizar para tomar decisiones propias, como por ejemplo la selección de ruta. Esta estimación se realiza para cada paquete recibido.

- **Selección de frecuencia.** La capa física debe ser capaz de establecer el canal en el cual se realizan las operaciones en función de lo que le especifiquen las capas de nivel superior.

- **Envío y recepción de tramas de datos.** De acuerdo con el estándar 802.15.4, el sistema radio debe ser capaz de emitir con una potencia de -3 dBm o superior y debe tener una sensibilidad mínima de -20 dBm.

En la figura 3.2 se muestra el formato de trama 802.15.4 a nivel físico. Dicha trama está compuesta principalmente por una cabecera de sincronización (SHR), una cabecera con campo para indicar la longitud de la trama (PHR) y la carga útil ó *payload*, que lleva datos del nivel superior (MAC). La cabecera de sincronización consta de un preámbulo necesario para entrenar al demodulador y una secuencia que indica el comienzo de la trama. El tamaño de esta cabecera es dependiente de la modulación y banda de transmisión empleadas. Para el mencionado caso de O-QPSK con DSSS en la banda de 2,4 GHz, el

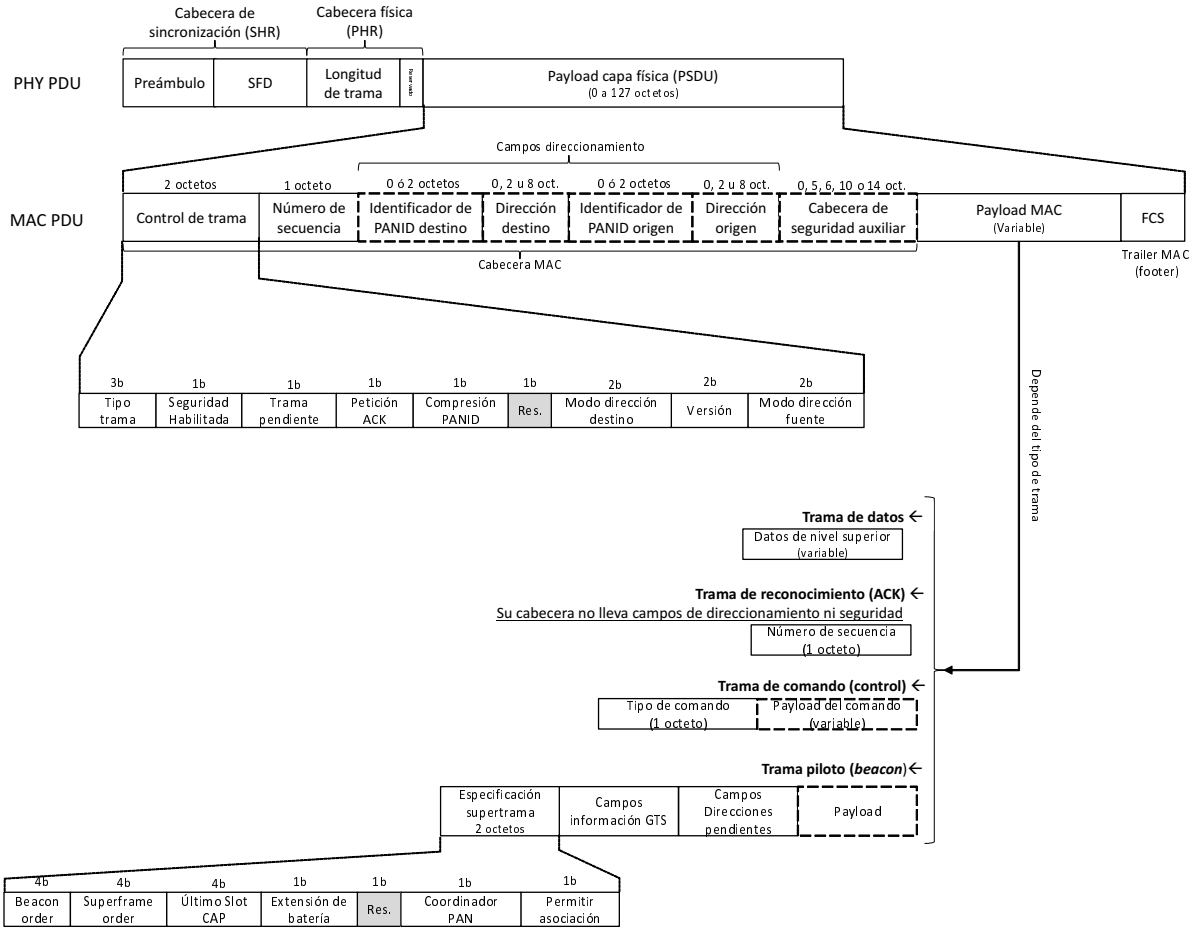


Figura 3.2: tramas o PDU (Protocol Data Unit) de nivel físico y MAC.

preámbulo es de 4 octetos (8 símbolos) y el delimitador de trama de un octeto (2 símbolos). El tamaño máximo del *payload* es de 127 bytes, estando limitado a 7 bits el campo para indicar la longitud. En la práctica, la longitud de este campo es de 5 octetos para las tramas MAC de ACK, y desde 9 a 127 octetos para el resto de tramas MAC.

La entidad de gestión de la capa física se encarga de mantener una base de datos, denominada PHY-PIB, con el estado de diferentes parámetros y atributos, algunos de los cuales pueden ser consultados (y en algunos casos establecidos) por los niveles superiores. Entre ellos se encuentran los parámetros que indican el canal en el que se va a operar (página e índice del canal), el modo de CCA, etc.

3.2.4. Capa MAC IEEE 802.15.4

La especificación de la capa MAC permite trabajar con las diferentes tecnologías de capa física permitidas por la norma, y puede interoperar con diferentes protocolos de nivel de red entre los que se incluye *ZigBee*.

La capa MAC es la encargada de establecer los mecanismos para que los diferentes dispositivos puedan hacer un uso compartido del mismo canal radio. El mecanismo principal escogido para ello es CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*, que establece que los dispositivos que quieran enviar información deben primero intentar

asegurarse de que el canal está libre y no hay una transmisión ya en marcha. La capa física es la que provee los mecanismos para realizar esta estimación (CCA). Una vez determinado que el canal no está ocupado, el dispositivo puede comenzar con la transmisión. Si el canal está ocupado, el dispositivo espera durante un tiempo aleatorio (periodo de *backoff*) antes de volver a intentarlo, pudiendo apagar la radio mientras tanto. Mientras el dispositivo no tenga acceso al canal, se repite este procedimiento hasta llegar a un número máximo de reintentos, en cuyo caso el mensaje a transmitir se descarta.

La capa MAC especificada por 802.15.4 tiene dos modos de funcionamiento relacionados con el refinamiento del mecanismo de control de acceso al medio: el modo balizado (*beacon-enabled*) y el modo no balizado (*nonbeacon*). En el modo balizado, el coordinador PAN (y coordinador *ZigBee*) se asegura de mantener una sincronización entre los dispositivos de la red mediante el envío de unos paquetes especiales denominados balizas, que pueden ser retransmitidos por los nodos coordinadores (*routers ZigBee*). La sincronización entre los nodos permite el establecimiento de intervalos temporales libres de contienda, en el que únicamente un dispositivo autorizado pueden transmitir, intervalos en los que los dispositivos pueden competir por el canal mediante el mecanismo CSMA/CA, y lapsos de tiempo en los que ningún dispositivo puede transmitir y por tanto todos pueden apagar la radio. En el modo no balizado no existe esta sincronización y por tanto el acceso al canal se realiza siempre mediante CSMA/CA. El modo balizado permite pues la existencia de periodos temporales (o *slots*) asignados a dispositivos, y podría permitir reducir el consumo de los nodos *routers* en algunas configuraciones, pero por contra tiene la desventaja de una mayor complejidad de implementación y de no adaptarse bien a la implementación de redes malladas. De hecho, en el caso de *ZigBee*, la norma específicamente indica que si se habilita el descubrimiento de rutas para permitir un funcionamiento en malla de la red, la capa MAC tiene que configurarse en modo no balizado.

Además de gestionar el acceso al canal y de mantener la sincronización con la red en el modo balizado, generando balizas en el caso de que el dispositivo sea un coordinador, la capa MAC también es responsable de gestionar los procedimientos mediante los cuales los dispositivos se agregan o abandonan la red (denominados procedimiento de asociación y desasociación respectivamente).

3.2.4.1. Modo balizado

En este modo el coordinador PAN transmite de forma periódica unos paquetes o tramas de gestión especiales denominados balizas (*beacon frames*), dando lugar a una estructura de envío de datos conocida como *supertrama* y que se muestra en la figura 3.3.

Una supertrama está delimitada por dos balizas consecutivas y se compone por un periodo activo, durante el cual el coordinador interactúa con la red, y un periodo inactivo durante el cual puede apagar la radio y entrar en bajo consumo. A su vez, el periodo activo se divide en un periodo de acceso por contienda (*Contencion Access Period, CAP*), durante el cual cualquier dispositivo que requiera transmitir datos puede acceder al canal utilizando el mecanismo CSMA/CA, y un periodo de acceso reservado (*Contention Free Period, CFP*), en el que únicamente pueden transmitir aquellos dispositivos a los que se les ha asignado previamente un intervalo dedicado para hacerlo. Para facilitar el control de

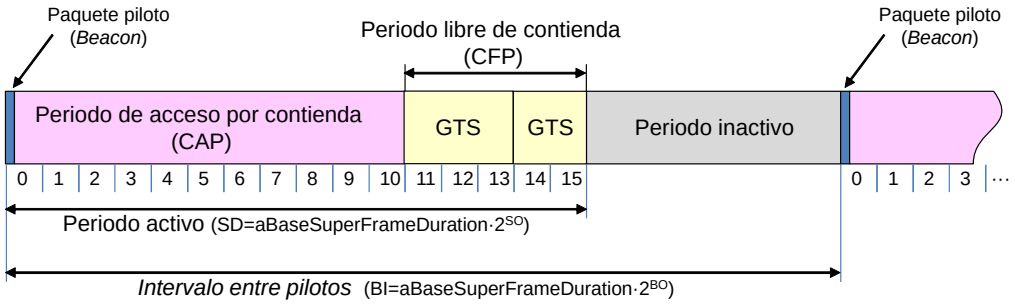


Figura 3.3: Estructura de supertrama.

la temporización, el periodo activo se divide en 16 *time slots* equiespaciados, numerados del 0 al 15, y que pueden repartirse entre los periodos CAP y CFP. La trama piloto o baliza enviada por el coordinador PAN marca el comienzo del primer *slot* y del periodo CAP, al cual sigue el periodo CFP. En este último tienen lugar, sin necesidad de aplicar el CSMA/CA, las comunicaciones a las que el coordinador PAN ha adjudicado uno o más *slots*. Un grupo de slots temporales asignados a un determinado nodo se denominan *Guaranteed Time Slots* (GTS). El periodo CFP es por tanto la suma de los diferentes periodos GTS asignados a distintas comunicaciones, pudiendo existir hasta un máximo de 7 de ellos. Mediante esta característica, el protocolo 802.15.4 puede ofrecer cierta calidad de servicio a determinadas aplicaciones con requerimientos especiales.

El coordinador PAN define la estructura de la supertrama en base a la configuración establecida por los protocolos de nivel superior al arrancar la red mediante diferentes parámetros:

- BO ($macBeaconOrder$). Determina, junto con la constante $aBaseSuperframeDuration$, la duración del intervalo de transmisión de las balizas, BI (*Beacon Interval*), mediante la siguiente expresión, en la que $0 \leq BO \leq 14$ y $aBaseSuperframeDuration$ es una constante de valor 960 que representa el número de símbolos que forman una supertrama de orden 0 (formada por 16 slots con una longitud de 60 símbolos cada uno).

$$BI = aBaseSuperframeDuration \cdot 2^{BO} simbolos \tag{3.1}$$

- SO ($macSuperframeOrder$). Determina la longitud del periodo activa de la supertrama (*Superframe Duration*, SD). El valor del SO y del SD están relacionados según la siguiente expresión, donde $0 \leq SO \leq BO \leq 14$:

$$SD = aBaseSuperframeDuration \cdot 2^{SO} simbolos \tag{3.2}$$

Si los valores de BI y SD coinciden la supertrama carecerá de periodo inactivo. Por otra parte, de acuerdo con el estándar 802.15.4, si el valor de configuración de SO y BO es 15 el coordinador PAN configurará la red para trabajar en modo no balizado. En una red configurada en modo balizado existe también la opción de que otros coordinadores,

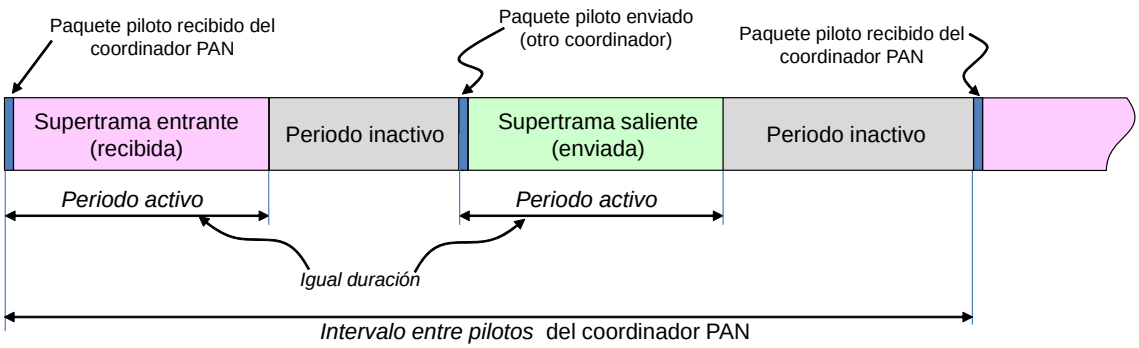


Figura 3.4: Supertramas de un coordinador PAN y un coordinador.

además del coordinador PAN, puedan enviar tramas piloto y crear su propia supertrama, siempre y cuando lo hagan durante el periodo inactivo del coordinador PAN. En este caso, la duración del periodo activo de la supertrama, debe ser la misma que la establecida por el coordinador PAN. La figura 3.4 muestra cómo sería la temporización desde el punto de vista del dispositivo coordinador. La supertrama del coordinador PAN se denomina en este escenario supertrama entrante (*Incomming Superframe*), y la supertrama generada por el otro dispositivo coordinador, supertrama saliente (*outgoing Superframe*), aunque en realidad en ellas se pueden dar tanto transmisiones como recepciones de datos por parte del dispositivo que la gestiona. La supertrama de un nodo coordinador no debe nunca solaparse con la del coordinador PAN, y en caso de que tal solape pudiera darse (por un cambio en la configuración), el dispositivo que no es coordinador PAN debería dejar de emitir sus balizas. Más allá de la preferencia del coordinador PAN sobre el resto de coordinadores, la norma IEEE 802.15.4 no establece ningún mecanismo que permita gestionar la sincronización entre diferentes coordinadores para evitar solapes cuando varios utilicen el modo balizado. Sin embargo, las últimas versiones del estándar sí que establecen que los niveles superiores pueden indicar al MAC en la primitiva de configuración de la supertrama (MLME-Start.request) el tiempo de separación (medido en símbolos) entre la supertrama de un coordinador y la de su coordinador padre. Como se comentará más adelante en la sección 3.2.5.2, la norma *ZigBee* especifica que en una red balizada se debe procurar que la supertrama de un dispositivo no se solape con la de sus vecinos ni con las de los padres de éstos, y establece algunos mecanismos para conseguirlo.

En el modo balizado, el coordinador que genera una supertrama puede configurar un modo de extensión de vida de la batería (*Battery Life Extension*, BLE), que permite desactivar la recepción durante el CAP una vez transcurrido un tiempo tras la transmisión de la baliza o paquete piloto. Este tiempo es un múltiplo entero de una unidad denominada periodo de *backoff*, que se definirá más adelante en la sección 3.2.4.4, y viene determinado por el parámetro de configuración $macBattLifeExtPeriods$, que indica cuántos periodos de backoff después de la transmisión de la baliza puede ser desactivada la recepción. Este comportamiento se ilustra en la figura 3.5.

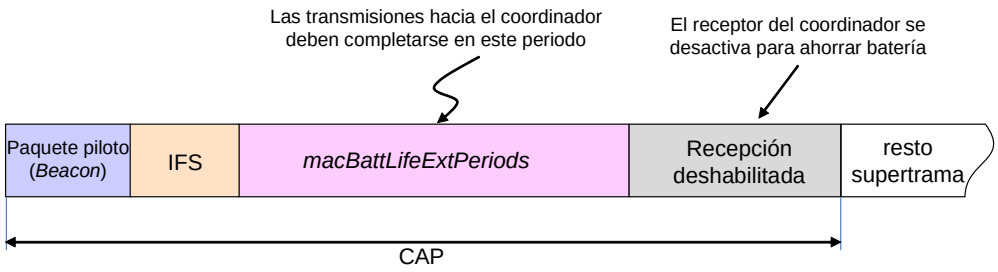


Figura 3.5: Extensión de vida de la batería.

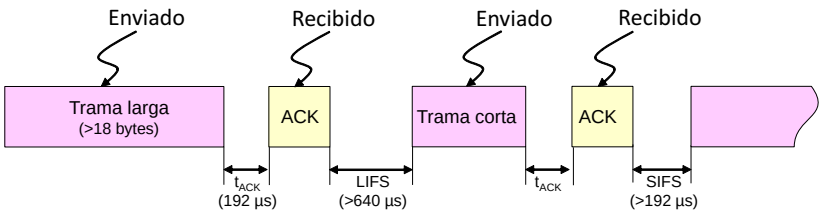
3.2.4.2. Modo no balizado

Como se ha comentado en el apartado anterior, la PAN queda configurada en modo no balizado cuando el valor de los parámetros BO y SO se establece a 15. En este modo ni el coordinador PAN ni ningún otro dispositivo coordinador emite balizas de forma regular, y no existe una estructura de supertrama ni ningún tipo de sincronización temporal entre los dispositivos, realizándose el control de acceso al medio siempre por contienda. Como se comentará más adelante en la sección 3.2.4.4, el algoritmo CSMA/CA utilizado en los modos balizado y no balizado es diferente, ya que en el primer caso, aprovechando la sincronización temporal proporcionada por la supertrama, se aplica CSMA ranurado (*Slotted CSMA*), y en el segundo caso CSMA no ranurado (*Unslotted CSMA*).

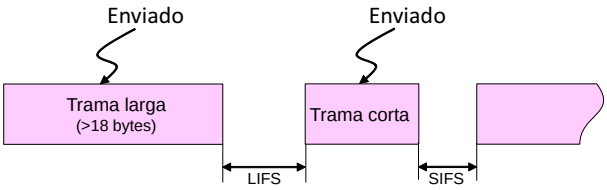
En modo no balizado no existe tampoco un periodo inactivo que permita a los nodos coordinadores (coordinadores y *routers ZigBee*) apagar la radio, motivo por el cual el consumo de los mismos aumenta considerablemente. No obstante, todavía es posible mantener bajo el consumo de los dispositivos finales gracias al mecanismo de transmisión indirecta, que se comentará con más detalle en la sección 3.2.4.5, y que sí que permite que estos dispositivos entren en modo de bajo consumo que únicamente abandonan ocasionalmente para transmitir datos o sondear a sus padres. Éstos en cambio tienen que permanecer continuamente con la radio activa, pues desconocen en qué momento despertarán los dispositivos finales.

3.2.4.3. Temporización de las tramas

Los paquetes o tramas transmitidos de un dispositivo a otro tienen que ser adecuadamente espaciados para dejar tiempo suficiente para que el dispositivo receptor los pueda ir procesando adecuadamente. En la norma se establece un tiempo mínimo entre la transmisión de dos paquetes consecutivos que depende del tamaño de los paquetes que se han enviado. Tras la transmisión de una trama pequeña (menor o igual a 18 octetos), se debe dejar un tiempo de al menos 12 símbolos antes de transmitir una nueva trama (es decir, 192 μs si la modulación empleada por la capa física es O-QPSK con DSS). Si la trama es grande (mayor de 18 octetos), entonces se deben esperar al menos 40 símbolos (640 μs con la misma modulación) antes de volver a transmitir. Estos tiempos de guarda se denominan SIFS (*Short inter-frame spacing*) y LIFS (*Long Inter-frame spacing*), respectivamente. En el caso de que la transmisión lleve confirmación, tras cada paquete recibido el dispositivo



a) Transmisión con confirmación.



b) Transmisión sin confirmación

Figura 3.6: Espaciado de los paquetes o tramas 802.15.4.

receptor debe a su vez enviar un paquete ACK en respuesta reconociendo la correcta recepción del paquete. En este caso, el envío del ACK debe realizarse 12 símbolos después de haber terminado la recepción del paquete (intervalo que recibe la denominación de t_{ACK}). Este tiempo coincide con un parámetro constante de la capa física que especifica el tiempo máximo que puede emplear la radio en conmutar de transmisor a receptor o viceversa ($aTunraroundTime$). El intervalo de espaciado entre tramas empieza a contarse en este caso tras la recepción del ACK. La figura 3.6 ilustra ambos casos.

3.2.4.4. Algoritmo CSMA/CA

El algoritmo CSMA/CA debe ser ejecutado por aquellos dispositivos que deseen transmitir en el periodo CAP de una red balizada o en una red no balizada, con la excepción de los paquetes ACK de confirmación, que se envían inmediatamente (respetando el tiempo de guarda, t_{ACK}) tras la recepción de un paquete previo. Tampoco lo ejecutan los dispositivos coordinadores cuando realizan el envío de las tramas piloto, y en algunos casos es posible omitirlo por parte de los coordinadores en el procedimiento de transmisión indirecta cuando se hallan en disposición de responder rápidamente a un sondeo realizado por el dispositivo final.

El algoritmo utiliza el procedimiento CCA ofrecido por la capa física para estimar si el canal está o no ocupado, lo cual debe verificarse antes de acceder al canal, intentando así evitar interferir en la comunicación de otros dispositivos. El CCA debe ser realizado durante al menos 8 símbolos (64 μs) para poder estimar adecuadamente si el canal está o no libre, según especificación de la capa física.

Como ya se mencionó anteriormente, se definen dos tipos distintos de algoritmo CSMA/CA según se esté utilizando el modo balizado o el no balizado. Para el modo balizado se emplea el CSMA/CA ranurado mientras que para el no balizado se utiliza el CSMA/CA no ranurado. En ambos casos los intentos de un transmisor para acceder al medio están limitados por un tiempo de espera pseudoaleatorio (*backoff*), que es siempre un múltiplo

entero de una unidad de tiempo denominada periodo de *backoff* y que es determinado por una constante de configuración de la capa MAC (*aUnitBackoffPeriod*, 20 símbolos). Cada vez que un dispositivo no obtiene acceso al canal por encontrarlo ocupado, realiza una nueva espera aleatoria antes de volver a intentar realizar la comprobación CCA. En el CSMA/CA ranurado, los límites de los periodos de *backoff* deben estar alineados con las balizas transmitidas por los coordinadores, y la capa MAC debe asegurarse que la transmisión del paquete por parte de la capa física comienza alieada con estos límites (nótese por tanto que las «ranuras» del CSMA/CA ranurado vienen marcadas por los periodos de backoff y no por los *slots* de la supertrama, que únicamente se utilizan para determinar la longitud del CAP). Para el caso del CSMA/CA no ranurado, cada transmisor mantiene su propia temporización. Para evitar posibles colisiones debido a la sincronización entre los dispositivos en el modo ranurado, debe comprobarse mediante el CCA que el canal permanece sin actividad durante varios periodos de *backoff* consecutivos (por defecto 2) antes de comenzar la tranmisión.

El algoritmo CSMA/CA utiliza tres variables para controlar el acceso al medio:

- NB (*Number of Backoffs*). Es el número intentos de acceso al canal que un dispositivo lleva acumulados y que se va incrementando cada vez que no se obtiene acceso al canal al realizar el CCA. Esta variable se inicializa a 0 cuando el dispositivo comienza a intentar el envío de un nuevo paquete. Si tras sucesivos intentos fallidos de obtener el canal esta variable supera el límite $macMaxCSMABackoff$ (un parámetro de configuración del MAC que puede valer entre 0 y 5, y cuyo valor por defecto es 4), se aborta la ejecución del algoritmo CSMA/CA y se informa de un fallo de acceso al canal a los niveles superiores.

- CW (*Contention Window*). Se utiliza únicamente en el modo balizado y sirve para controlar el número de periodos de backoff que el canal debe estar sin actividad para que pueda considerarse libre. Esta variable se inicializa a 2 cada vez que se va a realizar un nuevo intento de acceso al canal y se va decrementando cada vez que el CCA determina que el canal se encuentra sin actividad, realizándose la transmisión cuando llegue a 0 (en la práctica, se realiza el CCA en dos periodos de *backoff* consecutivos).

- BE (*Backoff Exponent*). Determina el rango del número aleatorio de periodos de *backoff* que el dispositivo debe esperar para volver a intentar acceder al canal después de un acceso fallido, y se va incrementando en cada acceso fallido para ir haciendo aumentar el rango de posibles valores a obtener, disminuyendo así las posibilidades de colisión. El tiempo de *Backoff* viene dado por la fórmula:

$$Backoff = rand(0, 2^{BE} - 1) \cdot aUnitBackoffPeriod \tag{3.3}$$

  Cada vez que se inicia una nueva operación del algoritmo CSMA/CA BE se inicializa al valor del parámetro de configuración $macMinBE$, que tiene un valor por defecto de 3 . En el caso del CSMA/CA ranurado, cuando el coordinador utiliza la extensión

de vida de batería, BE se inicializa como máximo a 2. Por otra parte, el parámetro $macMaxBE$ (cuyo valor por defecto es 5, aunque puede configurarse hasta un máximo de 8) determina el valor máximo hasta el que se puede incrementar BE (que permanece a dicho valor una vez alcanzado).

En el modo ranurado es importante tener en cuenta que todas las transmisiones realizadas mediante el mecanismo de contienda deben llevarse a cabo exclusivamente durante el CAP (y en el caso de tener activada la extensión de vida de la batería durante el número de periodos de backoff, $macBattLifeExtPeriods$ en los que el coordinador permanece activo. Si el MAC no puede asegurar que la transmisión puede llevarse a cabo antes del fin del CAP (incluyendo el *backoff*, la transmisión del paquete y la recepción del ACK en el modo confirmado), debe pausar la operación y reanudarla en el comienzo del siguiente CAP.

A continuación se incluye una descripción más formal de cada variante del algoritmo.

CSMA/CA ranurado Este algoritmo se divide en cinco pasos:

1. Inicialización de NB, CW y BE, $NB = 0$ y $CW = 2$. El valor de BE depende de la variable *macBattLifExt*. Si ésta es FALSE entonces $BE = macMinBe$, que por defecto vale 3. Si $macBattLifExt = TRUE$ (extensión de vida de la batería) entonces $BE = minimo(2, macMinBE)$.

2. Tiempo de espera aleatorio para intentar acceder al canal. Este tiempo de espera valdrá como máximo $2^{BE} - 1$ periodos de backoff. Si $BE = 1$ entonces el tiempo de espera es nulo, con lo que se salta directamente al paso 3.

3. Evaluación de canal libre (CCA o *Clear Channel Assessment*). El dispositivo escucha el canal en el comienzo del siguiente periodo de *backoff* (que están alineados con las balizas). Si el canal no se encuentra en uso, se salta al paso 5. Si el canal resulta estar ocupado pasamos al paso 4.

4. CW es reiniciado a su valor original (2). NB se incrementa en 1. BE también se incrementa en 1 siempre que no sobrepasemos el valor de *macMaxBE* (valor por defecto 5 según el estándar). Si $NB > macMaxCSMABackoffs$ (también de valor 5) se considera que la transmisión ha fallado. Si no, volvemos al paso 2.

5. El canal no está siendo usado. Se disminuye en 1 el parámetro CW. Si CW es 0 entonces se ha superado la ventana de contienda y el canal se considera libre. La transmisión puede comenzar. Si CW no es 0 todavía no hemos superado la ventana de contienda con lo que se vuelve al paso 3.

CSMA/CA no ranurado Este algoritmo es muy similar al visto en el punto anterior. Los pasos a seguir son en este caso 4:

1. Inicialización de NB, CW y BE. Al igual que en el caso anterior $NB = 0$. En este caso sin embargo $CW = 0$, con lo que no existe ventana de contienda: si se detecta

una única vez que el canal no está siendo usado se considera que está libre. Por otro lado, el modo no balizado no soporta el modo $macBattLifExt$, por lo que directamente BE se iguala a $macMinBE$.

2. Tiempo de espera aleatorio para intentar acceder al canal. Este tiempo de espera valdrá como máximo $2^{BE} - 1$ periodos de *backoff*. Si $BE = 1$ entonces el tiempo de espera es nulo, con lo que se salta directamente al paso 3.

3. Evaluación de canal libre (CCA o *Clear Channel Assessment*). El dispositivo escucha el canal al menos durante 8 símbolos (determinado por la capa física). Si el canal no se encuentra en uso, la transmisión puede comenzar. Si el canal resulta estar ocupado pasamos al paso 4.

4. NB se incrementa en 1. BE también se incrementa en 1 siempre que no sobrepasemos el valor de macMaxBE (valor por defecto 5 según el estándar). Si $NB > macMaxCSMABackoffs$ (también de valor 5) se considera que la transmisión ha fallado. Si no, volvemos al paso 2.

3.2.4.5. Servicios de la capa MAC

Como se ilustró en la figura 3.1, la Capa MAC ofrece al nivel superior un servicio de transporte de datos accesible a través del punto de acceso de subcapa común (*MCPS-SAP*), y además ofrece otro punto de acceso a servicios de gestión (*MLME-SAP*) que permite a los niveles superiores establecer parámetros de configuración de la capa MAC y realizar determinadas operaciones relacionadas con la gestión de la red. No todos los dispositivos tienen por qué implementar todas las capacidades que puede ofrecer la capa MAC de 802.15.4, los dispositivos de funcionalidad reducida RFD pueden no implementar algunas capacidades de los servicios de transporte de datos y gestión, e incluso algunas capacidades del servicio de gestión son de implementación opcional para los dispositivos de funcionalidad completa FFD (en particular, aquellas relacionadas con el establecimiento y mantenimiento de una supertrama, así como con la gestión de los GTS, cuyo soporte no es obligatorio).

Transporte de datos El servicio de transporte de datos permite a los niveles superiores transportar sus PDU de control o de datos entre dispositivos interconectados. Los paquetes enviados por los niveles superiores son almacenados en un *buffer* para su transmisión, e identificados mediante un manejador para su gestión. Este manejador será utilizado para confirmar su envío, indicar algún error en su tratamiento o para que su transmisión pueda ser cancelada por el nivel superior antes de que se llegue a producir (esto último es optativo en los RFD).

El procedimiento mediante el que se lleva a cabo el transporte de datos depende del escenario y de algunas opciones de configuración, entre las que destacan:

- Transmisión durante el CAP o durante un GTS. En el primer caso se debe aplicar el algoritmo CSMA/CA, mientras que en el segundo, no.

- Transmisión directa o indirecta. La transmisión directa se lleva a cabo en una comunicación *peer-to-peer* entre dispositivos coordinadores (routers y coordinador ZigBee) y en el envío de datos desde un dispositivo final hacia un dispositivo coordinador (es decir, desde un dispositivo final hacia un *router* o un coordinador). La transmisión indirecta sólo puede ser realizada por los coordinadores (*routers* o coordinador *ZigBee*) y se utiliza en el envío de información desde éstos hacia los dispositivos finales, con el objetivo de reducir el consumo de los últimos. En el mecanismo de transmisión indirecta, el paquete no se intenta enviar por el coordinador inmediatamente tras estar disponible, sino que se almacena en una cola de transmisión a la espera de que el dispositivo final envíe un paquete de sondeo (*Poll*) hacia el coordinador. Esto permite que el dispositivo final pueda tener la radio apagada, y sólo encenderla en el momento de realizar el sondeo para extraer datos del dispositivo coordinador. El comportamiento difiere según el modo:

  - En el modo balizado, los dispositivos coordinadores indican en las tramas piloto la presencia de datos destinados a otros dispositivos. Esta información puede ser utilizada por los dispositivos finales para enviar un mensaje de sondeo al coordinador (aplicando CSMA/CA) e indicarles que están disponibles para recibir datos. Una vez recibido el mensaje del dispositivo el coordinador envía un ACK de confirmación, y posteriormente los datos dirigidos hacia el dispositivo final, en este caso sin aplicar CSMA/CA si se puede garantizar que se hace lo suficientemente rápido tras recibir la petición (en menos de un periodo de backoff más un SIFS).
  - En el caso no balizado, los dispositivos finales despertarán de forma más o menos periódica para comprobar si tienen transacciones indirectas pendientes, mediante el envío de un paquete de sondeo hacia su coordinador padre. El paquete de sondeo está sujeto a la aplicación del CSMA/CA no ranurado, y debe ser confirmado por el dispositivo padre mediante el envío de un paquete de ACK. Si no hay datos pendientes para ser enviados hacia el dispositivo final, el coordinador lo puede indicar en el mensaje de confirmación, aunque también puede hacerlo mediante el envío de una PDU de datos de tamaño 0. Si hay datos disponible, el paquete de datos debe enviarse, en este caso aplicando el mecanismo CSMA/CA.

  Tanto en un modo como en otro, el dispositivo destino debe permanecer con la recepción activada durante un tiempo $macMaxFrameTotalWaitTime$, que se fija (medido en símbolos) en base a las configuración de las capas MAC y física, y que básicamente se deriva de la suma del tiempo máximo de transmisión de una trama y del caso peor de espera en el que se produzcan el máximo número de *backoffs* y el valor aleatorio del mismo tome el valor más alto en todos ellos. Si no se recibe ningún mensaje desde el dispositivo coordinador transcurrido este tiempo, el dispositivo final entenderá que no había datos disponibles y volverá a entrar en bajo consumo. Si el paquete se recibe de forma correcta, uno de sus campos puede indicar si quedan más tramas pendientes almacenadas en el coordinador, de

forma que el dispositivo final pueda solicitar inmediatamente su envío mediante un nuevo procedimiento de sondeo. La transacción indirecta tiene un límite de tiempo $macTransactionPersistenceTime$ que puede permanecer el paquete en la cola de transmisión del coordinador sin ser satisfactoriamente extraído por el dispositivo final, transcurrido el cual se descarta y se genera una indicación al nivel superior (*TRANSACTION EXPIRED*). La realización del sondeo por parte de los dispositivos esclavos es iniciada por los niveles superiores.

- Transmisión con o sin confirmación. En el modo con confirmación, el dispositivo destino debe transmitir de vuelta un paquete ACK confirmando la correcta recepción del paquete si ésta se ha producido. Esta confirmación es opcional y debe ser solicitada por el dispositivo origen mediante un campo de control de la trama MAC. El envío con confirmación es solicitado por la capa de nivel superior a la capa MAC en la primitiva de petición de envío. Cuando el paquete se envía con confirmación, el dispositivo origen debe esperar un tiempo $macAckWaitDuration$ hasta recibir el ACK. Si se recibe, se elimina del buffer y se informa al nivel superior de la correcta operación. Si transcurrido dicho tiempo no se recibe la correcta confirmación, se entiende que el intento de transmisión ha fallado y debe continuar intentándolo. En este punto, se distingue si se trata de una transmisión directa o indirecta.

    - Si es una transmisión directa, el dispositivo repetirá el intento de transmisión hasta un máximo de $macMaxFrameRetries$ reintentos (siempre respetando los límites temporales de la transmisión y aplicando el algoritmo CSMA/CA). Si se supera el máximo de intentos el paquete se descarta y se informa al nivel superior mediante una primitiva de que no se ha conseguido confirmar el correcto envío del paquete (*NO ACK*). $macMaxFrameRetries$ debe estar comprendido entre 0 y 7, con un valor por defecto de 3.
    - Si es una transmisión indirecta en la que falla la confirmación del envío del paquete con datos desde el coordinador hasta el dispositivo final, el coordinador no intentará volver a enviarlo inmediatamente, sino que esperará a volver a ser sondeado por el dispositivo final, manteniendo el paquete en la cola de transmisión. En este caso el paquete se descarta por el coordinador cuando transcurre el $macTransactionPersistenceTime$ anteriormente mencionado, generando una indicación al nivel superior (*TRANSACTION EXPIRED*).

En la figura 3.7 se ilustra el funcionamiento de los diferentes escenarios de transporte de datos. Por simplicidad, en ella se muestra el caso ideal (donde no hay pérdidas de información y la transacción se realiza con éxito). Además no se detalla el procedimiento de CSMA/CA, que debe llevarse a cabo necesariamente antes de realizar la transmisión del paquete de petición de datos, y (salvo en casos puntuales) al enviar el paquete de datos.

Inicialización y mantenimiento de PAN Como hemos mencionado, la entidad de gestión de la capa MAC es la encargada de controlar la conexión y desconexión de disposi-

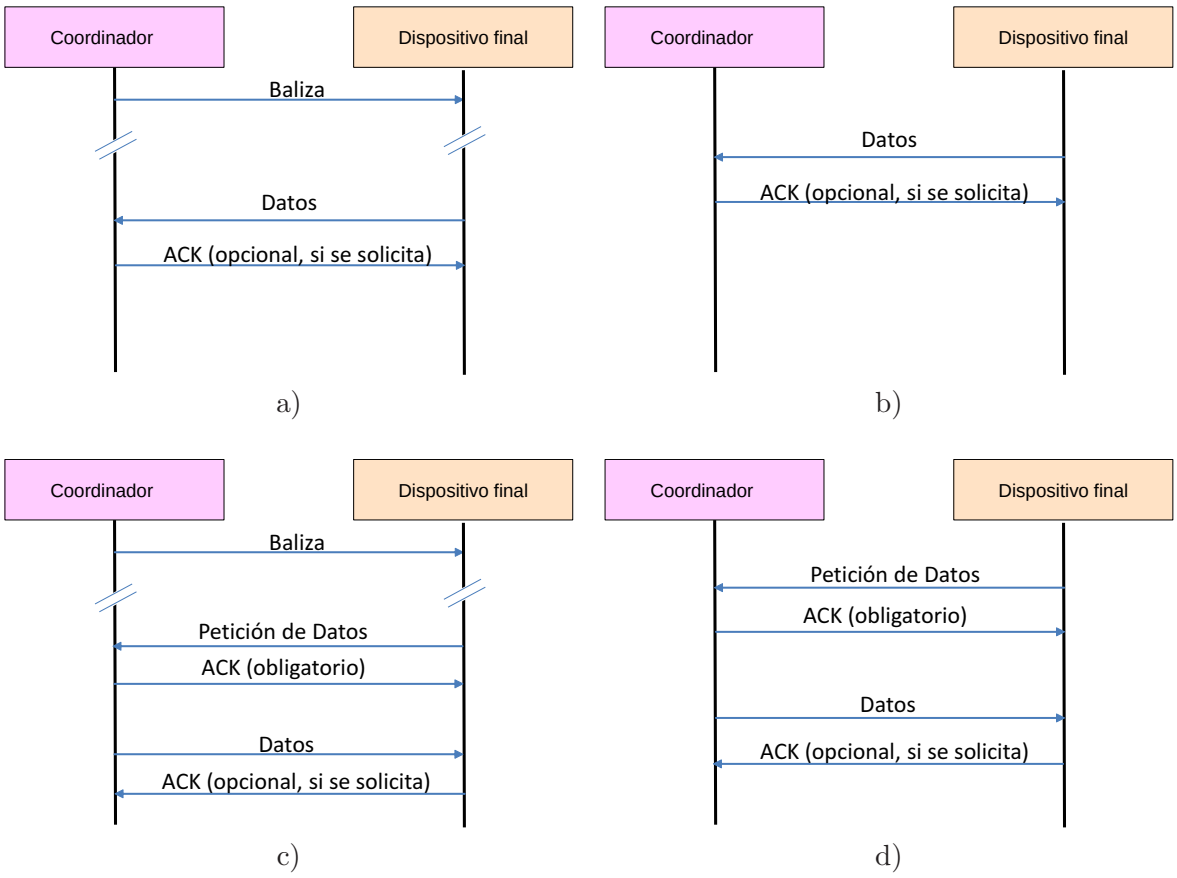


Figura 3.7: Transportes de datos directo (a y b) e indirecto (c y d), en modo balizado (a y c) y no balizado (b y d).

tivos a la red (procedimientos llamados asociación y desasociación en 802.15.4), así como de iniciar la operación de la red en el caso del coordinador PAN. También es la encargada de gestionar otros procedimientos de mantenimiento, como pueden ser la reserva de *slots* garantizados GTS, el establecimiento y modificación la estructura de la supertrama, y la recuperación de los dispositivos que hayan perdido sincronización con la red.

La realización de operaciones de gestión puede requerir el intercambio de mensajes de control entre las entidades MLME de diferentes dispositivos, lo cual se lleva a cabo mediante PDUs (o tramas) de comando, sobre los que se proporcionarán algunos detalles adicionales en la sección 3.2.4.6. El envío de estas tramas de comando también está sujeto a los mismos mecanismos de control de acceso al medio y restricciones que las tramas de datos.

Para el establecimiento de una nueva red, así como para la detección de redes ya existentes y para la resolución de conflictos de identificador PAN (PANId), la capa MAC se sirve de los siguientes procedimientos:

- Detección de energía de canal. Mediante este procedimiento, y a petición de las capas superiores, la capa MAC mide la energía de un canal o lista de canales determinados. Para ello indica a la capa física realizar una detección de energía de canal (ED) para cada uno de los canales elegidos. Este procedimiento es utilizado por el coordinador PAN durante la inicialización de red para seleccionar el canal con menor nivel (aparente) de interferencia entre los varios posibles.

- Escaneo activo de canal. Permite a un dispositivo localizar a un coordinador que esté operando dentro de su rango de alcance. En una red *ZigBee*, los *routers* y los dispositivos finales usan este procedimiento a la hora de asociarse a la red. En un coordinador *ZigBee* este método se puede utilizar para detectar las redes PAN que operan en el mismo canal dentro del radio de alcance, para así seleccionar un identificador distinto a los ya existentes para la nueva red. Para ello, el dispositivo que realiza el escaneo activo transmite PDUs de control con el comando se solicitud de baliza. Si un coordinador cercano trabaja en modo no balizado y recibe esta PDU, responderá inmediatamente con el envío de una baliza. En cambio, si el coordinador trabaja en modo balizado, ignorará la PDU de petición y seguirá transmitiendo de forma regular sus balizas. En cualquier caso, el dispositivo que realiza el escaneo recibirá una serie de balizas que le permitirán descubrir las redes que hay en la vecindad.

- Escaneo pasivo de canal. La única diferencia con el caso anterior es que el dispositivo que realiza el escaneo únicamente escucha en el canal a la búsqueda de posibles balizas transmitidas por dispositivos coordinadores cercanos, y que pueden indicar la existencia de una PAN en la vecindad. Obviamente, este modo es menos apropiado para detectar redes no balizadas, ya que en éstas no se transmiten balizas de forma regular.

- Escaneo de canal por un dispositivo huérfano. Si el dispositivo determina que ha perdido la sincronización con su coordinador emitirá lo que se conoce como notifi-

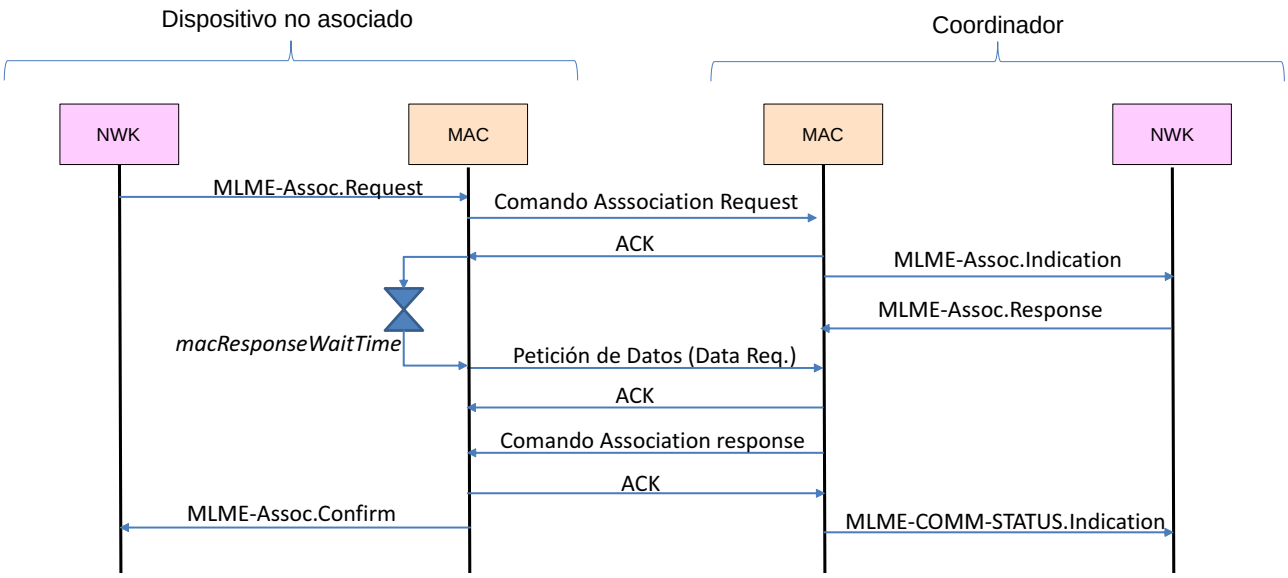


Figura 3.8: Procedimiento de asociación a un coordinador.

caciones de orfandad para intentar reencontrar la PAN con la que está actualmente asociado. Normalmente esta es una decisión tomada por las capas superiores cuando se detectan fallos reiterados en la comunicación. Las capas superiores indicarán a la capa MAC por qué canales transmitir dichas notificaciones. El coordinador debe responder a dicha notificación mediante una PDU de control con el comando de realineamiento para intentar que la sincronización entre él y el dispositivo huérfano se recupere.

El procedimiento de asociación es aquel por el cual un dispositivo no asociado se agrega a la red, asociándose a un coordinador que queda como su dispositivo padre. El procedimiento se muestra en la figura 3.8, y es iniciado mediante una primitiva por la capa de red de nivel superior. Para poder asociarse a la red es necesario haber descubierto previamente al coordinador, mediante los procedimientos comentados anteriormente, y que éste esté configurado para aceptar asociaciones. El dispositivo que desea asociarse comienza por enviar (sujeta como siempre al control de acceso al medio CSMA/CA) una PDU de comando de petición de asociación (*Association Request*), que será confirmada por el coordinador en caso de recibirse correctamente. Esta confirmación no implica que se haya aceptado aún la asociación,únicamente confirma que la petición se ha realizado correctamente. La petición es comunicada al nivel superior del coordinador mediante una primitiva de indicación de la entidad de gestión del MAC. Si se acepta la asociación, se enviará una PDU con el comando *Association response* mediante transferencia indirecta desde el coordinador hacia el dispositivo. El dispositivo debe esperar un tiempo $maxResponseWaitTime$ antes de sondear al coordinador para recibir la PDU mediante transferencia indirecta, para dar tiempo a que la respuesta esté disponible. Una vez recibida la respuesta, si es conforme, la asociación queda establecida y se notifica a los niveles superiores.

El procedimiento para eliminar un dispositivo de la red de forma controlada se denomina desasociación, y se muestra en las figuras 3.9 y 3.10. La desasociación puede ser

iniciada por el coordinador o por el propio dispositivo, dando lugar a dos escenarios ligeramente distintos. En ambos casos se realiza mediante el envío de una PDU con comando *Disassociation Notification* por parte del dispositivo que inicia el procedimiento. La diferencia radica en que esta PDU se transmite de forma directa desde el dispositivo hacia el coordinador, y mediante transferencia indirecta si va desde el coordinador al dispositivo.

El último procedimiento que se va a detallar es la notificación de orfandad (*Orphaning Notification*). Los dispositivos huérfanos son aquellos que estaban asociados a una red y sin notificación previa pierden contacto con su dispositivo coordinador. Esta pérdida de contacto puede ser detectada por las capas de nivel superior cuando detectan fallos de comunicación repetidos. Como ya se mencionó, los paquetes que son enviados por el MAC con confirmación se intentan retransmitir un número máximo de veces $macMaxFrameRetries$ (3 por defecto) antes de descartarlo. Si se supera este número, se notifica un fallo de comunicación al nivel superior. Tras uno o varios de estos fallos, el nivel superior puede declarar el dispositivo huérfano e iniciar el procedimiento de notificación de orfandad para intentar volver a contactar con su coordinador. Es de destacar que la norma 802.15.4 deja la responsabilidad de fijar el valor del número de fallos que deben producirse para dar al dispositivo por huérfano a los desarrolladores de los niveles superiores, y por su parte el núcleo del estándar *ZigBee* no recoge ninguna recomendación al respecto. El procedimiento se muestra en la figura 3.11, y básicamente consiste en el escaneo de canal de dispositivo huérfano anteriormente mencionado, en el cual se envían por diferentes canales PDUs con el comando *Orphan Notification* y se espera respuesta por el mismo canal durante un periodo de tiempo $macResponseWaitTime$. El nivel MAC de los coordinadores que reciban esta PDU contactará con el nivel superior del mismo nodo para comprobar si el dispositivo estaba en efecto previamente asociado. En caso afirmativo, enviarán una PDU con el comando *Coordinator Realigment* para informar de la configuración y estado de la red y permitir que el dispositivo pueda resincronizarse con el coordinador.

3.2.4.6. Formato de trama MAC

El formato de las PDU de nivel MAC se muestra en la figura 3.2 junto con el formato de la PDU de nivel físico. Algunos campos sólo están presentes en algunos tipos específicos de PDU y otros son completamente opcionales. La capa MAC distingue cuatro tipos principales de tramas:

- **PDUs de datos.** Se utilizan para el transporte de paquetes de datos de nivel superior.
- **PDUs de comando.** Se utilizan en las operaciones del plano de control, para intercambiar órdenes entre las entidades MLME de diferentes dispositivos en los diferentes procedimientos de gestión de la red. Algunas las hemos mencionado previamente al comentar estos procedimientos (asociación, desasociación, etc.).
- **PDUs de confirmación (ACK).** Se utilizan para confirmar la correcta recepción de un paquete de datos o de control, y no llevan *payload*.

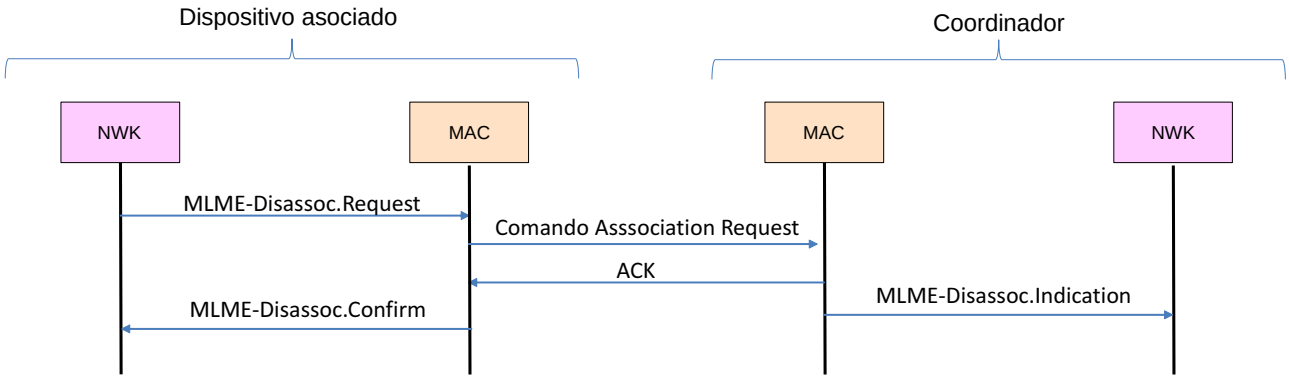


Figura 3.9: Procedimiento de desasociación arrancado por el dispositivo.

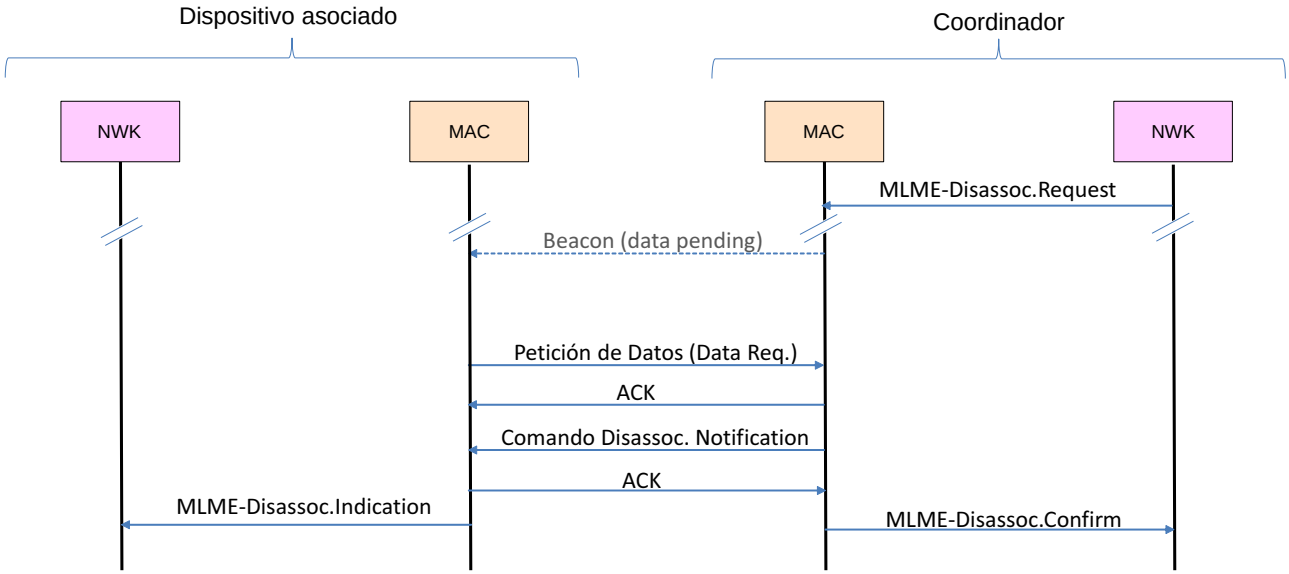


Figura 3.10: Procedimiento de desasociación arrancado por el coordinador.

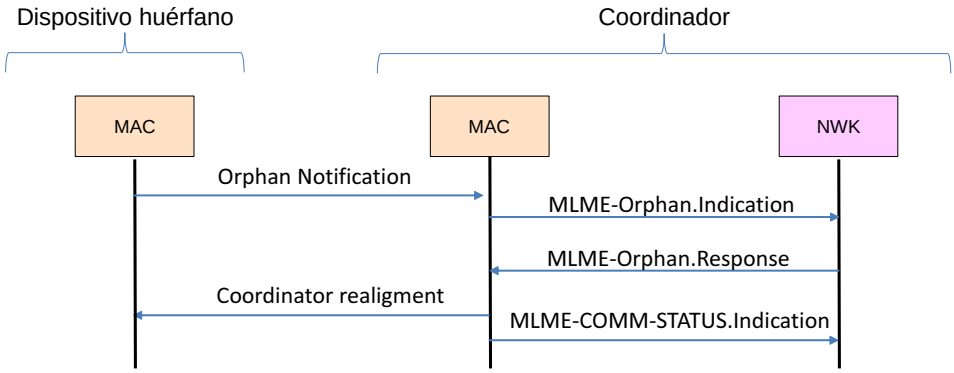


Figura 3.11: Procedimiento de *orphaning* o notificación de orfandad.

- **Pilotos o balizas.** Se utilizan para la detección de redes PAN en la vecindad y para mantener la sincronización de la supertrama (e indicar la disponibilidad de datos de transferencia indirecta) en el modo balizado.

De forma general, las tramas o PDU de nivel MAC están tramas están compuestas por:

- **Cabecera (MHR o MAC Header)**, que básicamente permite identificar el tipo de PDU y desde donde y/o hacia quién va dirigida, así como obtener algunos parámetros relativos a la seguridad utilizada. Este campo a su vez se divide en varios:

  - **Control de trama:** Es un campo de 16 bits que permite indentificar el tipo de trama, y aporta información adicional necesaria para su decodificación, como por ejemplo el formato y/o presencia de los diferentes campos de direccionamiento, o si la seguridad está o no habilitada. Además, permite realizar algunas indicaciones como por ejemplo si se solicita confirmación de su recepción por parte del nodo destino, si el nodo que la envía tiene información adicional pendiente de ser transferida de forma indirecta hacia el destino, etc.
  - **Número de secuencia:** 8 bits para identificar la trama y distinguirla de las precedentes y sucesivas.
  - **Campos de direccionamiento:** Son varios campos que pueden los nodo origen y destino de la PDU. Su presencia y codificación dependen de lo indicado en el campo de control de trama. En principio se trata de 4 posibles campos :
    - Identificador de PAN destino: 16 bits para identificar el número de PAN con el que nos estamos comunicando. Se utiliza el valor hexadecimal FFFF para indicar "cualquier PAN".
    - Dirección destino: Puede ser la dirección MAC larga de 64 bits, que es virtualmente *única en el mundo* y se asigna en fábrica a cada dispositivo, o la dirección PAN de 16 bits, única en la red y que se obtiene al asociarse.
    - Identificador de PAN origen (16 bits).
    - Dirección origen: De nuevo, Puede ser la dirección MAC larga de 64 bits, que es virtualmente *única en el mundo* y se asigna en fábrica a cada dispositivo, o la dirección PAN de 16 bits, única en la red y que se obtiene al asociarse.

    Las PDUs de confirmación (ACK) no llevan estos campos para reducir su tamaño y por consiguiente el tiempo para crearlas y enviarlas. Las balizas sólo llevan información sobre la dirección origen, ya que no van dirigidas a ningún destino.

- ***Payload* o Carga útil.** Este campo varía para cada uno de los cuatro tipos de tramas. Las tramas de confirmación (ACK) tampoco contienen este campo. En la figura 3.2 se intenta mostrar la estructura del *payload* de cada una de las tramas:

- **PDUs de datos:** El *payload* coincide con los datos de nivel superior a transmitir.
- **PDUs de comando:** llevan un primer campo de 1 octeto con un código que identifica el comando de control, y un campo de longitud variable con los datos del mismo (y cuyo contenido depende del tipo de comando de control).
- **Balizas:** LLevan un campo con información de especificación de la supertrama (Beacon orde, supreframe order, fin del periodo CAP, etc.), y un conjunto de campos con listas de GTS y listas de dispositivos que tienen transacciones indirectas pendientes. También pueden llevar algunos datos de nivel superior que la capa de red haya solicitado enviar en las balizas, para completar la información que éstas proporcionan sobre la red.

- **Secuencia de control (MFR o MAC Footer):** secuencia de 16 bits conocida como FCS (Frame Check Sequence) que no es más que un código CRC (código de redundancia cíclico).

3.2.5. Capa de red ZigBee

La capa de red amplía la funcionalidad de la capa MAC en la configuración, formación y mantenimiento de la red PAN, así como en el transporte de datos por la misma, permitiendo la comunicación de los niveles superiores de dispositivos sin conexión directa entre ellos.

Básicamente las tareas que realiza la capa de red son las siguientes:

- **Configuración de dispositivo.** Permite comenzar a operar como coordinador *ZigBee* o unirse a una red ya existente como *Router* o *End-Device*.
- **Inicialización de la red PAN** (en el caso del coordinador).
- **Asociación, reasociación y desasociación** de una red.
- **Adjudicación de direcciones de red.** El nivel de red gestiona la asignación de las direcciones cortas de 16 bits a los dispositivos. Esta dirección debe ser comunicada al nivel MAC, que también puede utilizarla en sus operaciones.
- **Descubrimiento de la topología de red.** Consiste en la búsqueda de vecinos y rutas, y sólo la realizan los dispositivos *routers* y coordinadores.
- **Encaminamiento (routing).** La capa de red es capaz de gestionar envíos *unicast*, *broadcast* o *multicast* para el intercambio de datos entre cualquiera de los diferentes dispositivos de la PAN. Sólo los dispositivos *routers* y coordinadores pueden redirigir paquetes de red.

A nivel de red, existen dos tipos de direcciones: direcciones cortas (16 bits) y direcciones largas o direcciones IEEE (64 bits). La dirección IEEE se asigna a cada dispositivo en fábrica, y al menos en la práctica se puede considerar que es única en el mundo para

dicho dispositivo. Por contra, la dirección corta es asignada por la capa de red de forma dinámica. Dentro de una red *ZigBee* no debe haber más de un dispositivo con igual dirección corta. Como ya se ha mencionado previamente, aunque tanto el nivel MAC como el nivel de red utilizan ambas direcciones, la configuración de la dirección IEEE de un dispositivo es establecida por la capa MAC, mientras que la asignación de la dirección corta es responsabilidad del nivel de red.

3.2.5.1. Gestión de la red y de los dispositivos

En esta sección de describen con cierto nivel de detalle los procedimientos que ofrece la capa de red para iniciar la operación de una nueva red PAN, realizar la búsqueda de redes PAN en la vecindad, agregarse o abandonar una red ya existente, y asignar direcciones a los dispositivos.

Establecer una nueva red. Sólo los dispositivos con capacidad para comportarse como coordinador y que no están actualmente asociados a una red pueden realizar este procedimiento.

El primer paso para establecer una nueva red es encontrar un canal lo más idóneo posible para la operación. Esto se lleva a cabo utilizando los servicios de la capa MAC para realizar un escaneo de energía en un conjunto de canales, que permita descartar aquellos con un nivel de interferencia elevado, seguido de un escaneo activo de canal para descubrir posibles redes PAN en la vecindad. Una vez completadas estas operaciones, se elige aquel canal con menor interferencia y en el que trabaje un menor número de redes *ZigBee*. Una vez determinado el canal, se escoge un identificador de red (PANId) de 16 bits que no esté siendo utilizado por otra red PAN en la vecindad (y distinto de 0xFFFF, que tiene el significado especial «cualquier red») y se indica a la capa MAC dicho valor.

A continuación el dispositivo que ha iniciado la red, el coordinador, se autoasigna la dirección corta 0x0000, y por último se indica a la capa MAC que la red ha sido establecida con éxito.

Descubrimiento de red Mediante este procedimiento la capa de red indica a la capa superior qué redes *ZigBee* están operando dentro del rango de alcance del dispositivo en cuestión. El procedimiento se lleva a cabo preferentemente mediante la solicitud de un escaneo activo de canal a la capa MAC, aunque también puede hacerse mediante un escaneo pasivo.

Una vez completado, el procedimiento entrega al nivel superior de aplicación una lista con las direcciones PANId de las redes encontradas, indicando si permiten o no a otros dispositivos unirse a ellas. Además se informa también sobre parámetros de configuración detectados, como el canal en el que operan, la versión de *ZigBee* que implementan, y los parámetros *beacon* y *superframe order*. La parte de esta información que corresponde al nivel de red, se manda el en *payload* de datos de nivel superior que soporta la trama piloto de la capa MAC.

Permitir a los dispositivos unirse a una red Sólo el coordinador y los *routers ZigBee* (que son los que pueden actuar como coordinadores 802.15.4) pueden aceptar la unión de otros dispositivos a la red. Para aceptar la asociación de otros dispositivos, la capa de red modificará el valor del atributo *macAssociationPermit* de la capa MAC. La capa de red puede modificar dicho valor de forma indefinida o temporalmente durante un periodo determinado.

Unirse a una red El procedimiento se denomina asociación, igual que en el caso de la capa MAC. Al arrancar el proceso mediante la correspondiente primitiva, el nivel superior indicará al nivel de red cómo debe agregarse el dispositivo a la PAN, si como *router* o como *end-device*.

El primer paso para unirse a una red es que el dispositivo que se quiere agregar (y que va a quedar como hijo de otro dispositivo *router* o coordinador que forme parte de la red) efectúe un procedimiento de descubrimiento de red. Una vez seleccionada la red en la que vamos a trabajar se realiza un listado con los posibles candidatos, que serán aquellos nodos que tengan activado el permiso para unirse a ellos y con el que se tenga un coste de enlace inferior a 3 (el coste de enlace es una métrica de la calidad del nivel de enlace, que se comentará más adelante, y que se deriva de la calidad del paquete recibido (LQI) medida por la capa física. Entre los posibles nodos se seleccionarán aquellos más cercanos en la jerarquía al coordinador, y de entre éstos, el que cuente con un menor número de hijos. Una vez seleccionado el dispositivo padre, se inicia la asociación con él mediante la correspondiente primitiva de nivel MAC.

Al recibir la indicación de petición de asociación, el nivel de red del dispositivo padre comprobará la tabla de vecindad para determinar si se trata de un nuevo nodo o de uno que vuelve a reconectarse. En caso de no encontrarse en dicha tabla, procederá a asignarle una nueva dirección corta, que será única en la red, y la almacenará en la tabla. En adelante ambos dispositivos, padre e hijo, utilizarán esta dirección corta para comunicarse, incluso a nivel MAC.

Cuando un dispositivo hijo pierde la conexión con su dispositivo padre, puede intentar reconectarse utilizando el procedimiento de orfandad (*orphaning*), o bien (especialmente si falla el anterior), una reasociación. Este último procedimiento sirve para intentar volver a la red (aunque sea con otro padre), y es idéntico al que acabamos de describir excepto por el detalle de que, aunque el dispositivo padre haya cambiado su configuración y ya no acepte a otros dispositivos unirse a él, sí permitirá reasociarse al dispositivo hijo puesto que no se trata de una nueva asociación.

La otra opción cuando el dispositivo hijo ha perdido la conexión con el dispositivo padre es solicitar a la capa MAC realizar un "escaneo de canal por dispositivo huérfano" para intentar reubicar al dispositivo padre. Cuando la capa MAC de un dispositivo huérfano realiza este procedimiento envía su dirección IEEE. Cuando el padre recibe dicha dirección comprueba que dicho dispositivo era su hijo y le responde indicando su antigua dirección corta, con lo que se considera que la conexión se ha recuperado.

Abandonar una red El procedimiento por el cual un dispositivo abandona una red se denomina desasociación, y se pueden producir dos casos distintos: el padre decide que un hijo debe abandonar la red o el hijo comunica al padre que quiere abandonar la red.

En ambos casos, el dispositivo que inicia el procedimiento debe informar al otro mediante el envío de una PDU de control de nivel de red con el comando *leave*. Si el nodo que inicia el procedimiento es el hijo y se trata de un *router*, esta PDU puede enviarse en modo *broadcast* para que su salida de la red sea conocida por otros dispositivos que puedan depender de él para el transporte de datos. En cualquiera de los casos, el dispositivo padre debe actualizar su tabla de vecindad de forma que el dispositivo eliminado no aparezca en ella.

3.2.5.2. Redes en árbol balizadas y planificación de las supertramas

La capa de red de *ZigBee* permite la formación de redes con topología en árbol y redes con topología mallada. En ambos casos los dispositivos *routers* y coordinadores *ZigBee* pueden reencaminar paquetes, y a nivel MAC pueden funcionar como coordinadores 802.15.4.

Cuando se usa la topología en árbol se crea una estructura jerárquica en la que el nodo raíz es el coordinador, y el resto de nodos son *routers* o dispositivos finales. El coordinador *ZigBee* y los *routers* pueden tener otros hijos a su cargo, que también pueden ser *routers* o nodos finales. Cuando un dispositivo actúa como padre de otro, a nivel MAC actúa como su coordinador 802.15.4. En esta topología, un dispositivo no puede comunicarse directamente con cualquier otro, sino que únicamente puede hacerlo con su padre y con sus hijos. Gracias a esta limitación, las redes con topología en árbol pueden utilizar el modo balizado de la capa MAC. En este modo, los dispositivos *routers* pueden establecer su propia supertrama para comunicarse con sus hijos.

Siguiendo (y ampliando) las recomendaciones de 802.15.4, la norma *ZigBee* especifica claramente que se debe tener cuidado en intentar no solapar la supertrama de un dispositivo con las de sus vecinos y con las de los padres de estos. Para ello, la capa MAC permite ajustar la separación temporal entre la trama de un dispositivo y la de su padre. La localización temporal de las supertramas de los dispositivos vecinos viene determinada por la recepción de sus balizas. Dichas balizas pueden llevar además un *payload* con datos de nivel de red, indicando la separación de la supertrama del dispositivo vecino con respecto a la de su padre. Esto haría posible también determinar la localización temporal de las supertramas de los padres de los vecinos, incluso aunque no sean directamente visibles por él. En base a esta información, el dispositivo *router* puede determinar en qué instante temporal ubicar su supertrama sin colisionar con sus vecinos. Obviamente, para que este mecanismo sea viable, la duración de la supertrama debe ser mucho menor que el tiempo entre balizas, lo cual además es lo lógico cuando se quiere reducir el consumo.

En una red con topología mallada, cualquier dispositivo *router* puede comunicarse directamente con otro *router* o con el coordinador, permitiendo la búsqueda de rutas más cortas entre ellos. En este caso, no es posible utilizar el modo balizado. Aunque se utilice la topología mallada, la forma en la que se realiza la formación de la red sigue estableciendo una jerarquía en árbol, en las que unos dispositivos padres actúan como coordinadores

802.15.4 de sus hijos. La diferencia es que se pueden producir comunicaciones *peer-to-peer* al margen de dicha jerarquía.

3.2.5.3. Mecanismo distribuido de asignación de dirección corta.

Cuando un dispositivo padre (coordinador o *router*) recibe la petición de conexión por parte de otro dispositivo (candidato a dispositivo hijo), el dispositivo padre debe asignarle una dirección corta. Este mecanismo ofrecido por la red *ZigBee* permite asignar una única dirección corta a cada dipositivo que participa en la red PAN, y a la vez permite formar un arbol con rutas por defecto que pueden utilizarse para la transmisión de información por la red. Este mecanismo se utiliza tanto si la red es en árbol como si la red es mallada, y en este último caso en mecanismo de encaminamiento de la información por el árbol se vé complementado por otros mecanismos adicionales que permiten la búsqueda de rutas alternativas más óptimas.

Este algoritmo se basa en la generación de subgrupos de direcciones. El coordinador elige la primera dirección libre, 0x0000. A continuación, asigna un subgrupo de las direcciones restantes a cada uno de sus hijos con capacidades de router. Cada uno de esos hijos tendrá como dirección propia la primera del subgrupo que le ha sido asignado, teniendo las restantes direcciones disponibles para repetir el proceso con los dispositivos hijos que se conecten a él. Una vez asignados estos subgrupos de direcciones a los routers, se asignan las direcciones a los dispositivos finales, comenzando por la primera dirección libre.

Para calcular el tamaño de estos subgrupos de direcciones se tienen en cuenta ciertos parámetros de la red definidos por el coordinador:

- L_m: La máxima profundidad de la red, establecida por el parámetro de configuración de la red $nwkMaxDepth$.
- C_m: El máximo número de hijos que un padre puede tener establecida por el parámetro de configuración de la capa de red $nwkMaxChildren$
- R_m: Máximo número de *routers* que un nodo padre (coordinador o router) puede tener como hijos, establecifo por el parámetro $nwkMaxRouters$

Definiendo para cada dispositivo *router* un parámetro d como su profundidad en la jerarquía de la red (número de saltos necesarios por el camino más corto hasta llegar al coordinador, $0 <= d <= L_m$), para establecer los grupos de direcciones que puede gestionar se define una función $Cskip(d)$ mediante la ecuación 3.4:

$$Cskip(d) = \begin{cases} 1 + C_m \cdot (L_m - d - 1) & \text{si } R_m = 1 \\ \frac{1+Cm-R_m-C_m \cdot {R_m}^{L_m-d-1}}{1-R_m} & \text{en otro caso} \end{cases} \quad (3.4)$$

Al primer *router* hijo que se le asocie a este nodo de nivel d, se le asignará la dirección inmediatamente consecutiva a la del nodo padre, y conforme se vayan asociado más hijos *routers*, se le irán asignando direcciones separadas $Cskip(d)$ del anterior hijo. Es decir, si i es el número de hijo de un *router* de un padre de nivel d (comenzando la numeración en 0) su direccion vendra dada por:

$$Direccion_{router\_hijo}(i) = Direccion_{padre} + CSkip(d) \cdot i + 1 \tag{3.5}$$

Según este algoritmo las direcciones comprendidas entre $Direccion_{router\_hijo}$ y $Direccion_{router\_hijo} + CSkip(d) - 1$ serían gestionadas por el *router* hijo y asignadas a sus descendientes. Mediante este rango un *router* puede determinar si una determinada dirección destino pertenece a uno de sus descendientes o no. Conociendo las direcciones de sus hijos también es capaz de determinar de cuál de ellos es descendiente la dirección destino. Esto permite establecer un algoritmo relativamente sencillo de encaminamiento de paquetes por la red: Un nodo de dirección A y nivel d sabe que un paquete con dirección B es para uno de sus descendientes si $A < B < A + Cskip(d-1)$ y sabe que la dirección del siguiente salto (el hijo a quien lo tiene que enviar para que lo siga reencaminando) es $A+1+floor((B-(A+1))/Cskip(d)) \cdot CSkip(d)$. Si el paquete no es para sus descendientes (ni para él), entonces debe redirigirlo hacia su padre para que este lo siga propagando por la red.

A los nodos de tipo dispositivo final se le asignan las direcciones consecutivas tras la última del rango del último posible hijo, es decir, al n-ésimo *end-device* hijo de un *router* de nivel d (comenzando la numeración en 0), se le asigna la dirección:

$$Direccion_{end-device}(n) = Direccion_{padre} + Cskip(d) \cdot R_m + n \tag{3.6}$$

Además de este mecanismo, que es utilizado por la versión *tradicional* de *ZigBee*, existe otro mecanismo que es el utilizado en la variante *ZigBee-PRO* que se introdujo en 2007. En *ZigBee PRO* la red es mallada y se utiliza un mecanismo estocástico de asignación de dirección, en el que todas las direcciones (excepto la del coordinador, que sigue siendo la 0x0000) se asignan de manera aleatoria. En este caso es necesario establecer un mecanismo para resolver conflictos de dirección en las ocasiones (normalmente no muy probables) en las que se produzcan. Esta situación es detectada por los *routers* o el coordinador al descubrir discordancias en las tablas de vecindad o de encaminamiento y debe ser notificada por boarcast a la red para que la situación se resuelva, cambiando los dispositivos de dirección mediante una reasociación.

Nótese por tanto que las direcciones cortas son efímeras y pueden cambiar, incluso en el mecanismo distribuido de asignación en árbol, ya que se puede producir una reasociación y por tanto un cambio de padre cuando se pierde el contacto con el padre anterior. La única dirección que no cambia y que realmente identifica al dispositivo es la dirección larga de 64 bits. Esto debe ser tenido en cuenta por el nivel de aplicación.

3.2.5.4. Encaminamiento

El encaminamiento es el proceso que determina la ruta por la cual los mensajes van a ser propagados desde el dispositivo origen hacia el destino, y es una de las funciones principales del nivel de red. Como ya se ha mencionado, en una *ZigBee* los nodos *routers* y coordinador son los responsables de realizar dicha propagación redirigiendo los paquetes entre ellos, y además, si la red es mallada se encargan de descubrir y mantener las posibles rutas o caminos que los paquetes pueden seguir para llegar a su destino. Los dispositivos

finales no redirigen información ni son capaces de descubrir rutas, sino que entregarán el mensaje a su dispositivo padre para que éste lo haga por ellos.

En una red en árbol, la ruta a seguir por los paquetes viene prefijada por el algoritmo anteriormente mencionado, que se denomina encaminamiento jerárquico. En la red mallada esta ruta por defecto sigue existiendo (excepto en *ZigBee-PRO*), pero pueden buscarse caminos alternativos más óptimos. La idoneidad de un camino viene determinada por el número de saltos, la calidad de los enlaces entre los nodos que lo componen, consideraciones energéticas, etc. Por simplicidad, se recurre al concepto de *coste del camino* para definir una métrica que determinar cuándo un camino es mejor que otro. El coste de un camino es la suma del coste de los enlaces que lo componen. A su vez, el coste de un enlace se define en base a la probabilidad de pérdida de paquetes p_l en el mismo como:

$$Cl = min\left(\left(round\left(\frac{1}{P_l^4}\right)\right), 7\right) \tag{3.7}$$

La probabilidad de pérdida de paquete puede ser estimada mediante diversos métodos en base a la potencia recibida, la relación señal ruido medida y las estadísticas obtenidas durante el funcionamiento de la red. No obstante, por simplicidad, en una implementación práctica lo normal es derivar directamente el coste del enlace del nivel LQI medido por la capa física, mediante una tabla de consulta precalculada. El nivel LQI es medido por la capa física para cada paquete recibido y proporciona una estimación de la relación señal-ruido.

Los nodos coordinador y *routers* de una red *ZigBee* crean y actualizan una serie de tablas para facilitar su operación: tablas de vecindad, tablas de encaminamiento y tablas de descubrimento de ruta.

- Tablas de vecindad: La tabla de vecindad de un dispositivo almacena información de cada uno de los dispositivos detectados dentro de su rango de alcance y es de utilidad en diferentes contextos. Como ya se ha mencionado, es utilizada en el proceso de asociación para determinar si el dispositivo que se asocia ya formaba parte de la red, y también puede utilizarse por un dispositivo cuando necesita buscar un nuevo padre. Una vez asociado el dispositivo, esta tabla se utiliza para almacenar parámetros relacionados con la calidad del enlace, tipo de relación y demás información referente a los dispositivos dentro de su rango de alcance. La tabla de vecindad debe ser actualizada cada vez que se recibe una trama proveniente del otro dispositivo. Los parámetros que se almacenan en esta tabla para cada uno de los distintos dispositivos detectados son: dirección IEEE, dirección corta, tipo de dispositivo, relación con dicho dispositivo, LQI, permiso de asociación y si ha sido seleccionado como candidato a padre, entre otros.

- Tablas de encaminamiento: Se utilizan para realizar la redirección de los paquetes, y permiten determinar el siguiente nodo al que hay que reenviar el paquete para que llegue a su destino. Contienen una entrada por cada dirección destino para la que el nodo conoce una ruta. Cada entrada almacena la dirección corta del destino y la dirección corta del nodo al que hay que redirigir los paquetes que vayan hacia dicho

destino. Las tablas de encaminamiento también se utilizan en el procedimiento de búsqueda de rutas, y por ello en cada entrada se almacenan otra serie de campos como el estado de la ruta (si está activa, en proceso de descubrimiento o confirmación, inactiva, etc.)

- Tablas de descubrimiento de ruta: Son utilizadas en el procedimiento de búsqueda de ruta que se comentará más adelante para almacenar temporalmente cierta información importante en su descubrimiento y selección. En cada entrada se almacena el identificador del paquete de control con el comando de búsqueda de ruta, la dirección corta del dispositivo que origina dicho descubrimiento, la dirección corta del dispositivo vecino a través del cual ha llegado la petición de búsqueda de ruta, el coste acumulado de la ruta desde el origen hasta el destino, el coste (si se conoce) del camino desde el dispositivo actual que mantiene la tabla hasta el destino de la ruta y el tiempo de vida que le queda a dicha entrada de la tabla.

El procedimiento de búsqueda de ruta es iniciado por el nivel de aplicación solicitándolo al nivel de red mediante una primitiva de petición, y puede ser *unicast* (desde una única fuente hacia un único destino), *multicast* (una fuente, varios destinos) o *many-to-one* (varias fuentes, un destino). El escenario *multicast* se implementa mediante algunas direcciones que tienen valores especiales y que identifican los grupos de *multicast* a los que los disppsitivos se pueden unir. En el escenario *many-to-one*, es el dispositivo sumidero (destino) el que ordena la operación y en tal caso no se indica dirección destino.

Descubrimiento de ruta *unicast* El procedimiento utilizado por *ZigBee* está basado en el protocolo AODV [PBRD03]. El dispositivo que arranca el procedimiento difunde por la red, en modo *broadcast*, una PDU de control de solicitud de búsqueda (*route request*). Este paquete contiene un campo con el identificador de la petición y un campo de coste de camino, y es propagado en modo *boadcast* por todos los dispositivos que lo reciben. El campo coste de camino vale 0 en origen, pero es incrementado por cada nodo que propaga el paquete sumándole el coste estimado del enlace por el que lo han recibido, de forma que conforme la PDU se va propagando por la red, en dicho campo se va acumulando el coste del camino seguido por cada copia de la PDU. En cada dispositivo *router* en el que se reciba una copia de esta PDU se comprobará si en la tabla de descubrimiento de ruta ya hay una entrada con el mismo identificador y dirección origen, y la creará o actualizará en consecuencia. Un dispositivo puede recibir copias de la misma PDU de búsqueda por diferentes caminos, en cuyo caso irá actualizando la entrada de la tabla para almacenar los datos del paquete (coste y vecino por el que llegó) con menor coste acumulado. Las sucesivas copias que lleguen con un coste acumulado mayor que la entrada almacenada en la tabla se descargan y no seguirán siendo propagadas por la red. Cada vez que se actualice una entrada de la tabla de descubrimiento de ruta, el *router* debe actualizar la tabla de encaminamiento para incluir el vecino por el que ha llegado la PDU de solicitud de búsqueda de ruta como siguiente salto hacia la dirección que originó la petición. La entrada de la tabla se marca como provisional (búsqueda en curso).

En última instancia, las PDUs de búsqueda de ruta difundidas por la red deberían llegar

también al nodo destino, informándole del coste acumulado de la ruta y de la dirección del primer salto de la misma en sentido inverso (el vecino que ha entregado el mensaje). Para confirmar que el camino encontrado funciona también a la inversa, el dispositivo destino responde entonces con el envío de una PDU de control con el comando *route reply*. Esta PDU lleva como dirección destino la del nodo vecino que se ha seleccionado como siguiente salto en el camino hacia el nodo origen de la búsqueda de ruta, y como *payload* lleva la dirección de dicho dispositivo origen y el identificador de la petición de búsqueda. Este paquete es validado en cada uno de los nodos del camino mediante las tablas de encaminamiento y de búsqueda de ruta y retransmitido al siguiente salto hacia el nodo origen almacenado en la tabla de encaminamiento. Finalmente ésta es modificada, eliminando la entrada temporal (búsqueda en curso) y sustituyéndola por el nodo vecino por el que se ha recibido la PDU *route reply* como el siguiente salto hacia el destino del procedimiento de búsqueda (y nodo origen del *route reply*). La PDU *route reply* recorre por tanto de forma inversa la ruta óptima encontrada, confirmando que es correcta y actualizando las tablas de encaminamiento de cada uno de los nodos de la ruta para incluir la información del siguiente salto hacia el destino.

Si la dirección destino de un procedimiento de búsqueda de ruta es un nodo hoja, el dispositivo encargado de enviar la PDU de *Route reply* es su nodo padre. Este será también el encargado de recibir los mensajes unicast que se envíen hacia el hijo y almacenarlos a la espera de ser recogidos por el mecanismo de transmisión indirecta.

Descubrimiento de ruta *multicast* Es similar al procedimiento anterior, pero se utiliza la dirección de un grupo *multicast* en lugar de la dirección de un nodo como destino. Los dispositivos que no pertenezcan al grupo *multicast* se comportarán al recibir la PDU de solicitud de ruta igual que en el procedimiento anterior, difundiéndola por la red y actualizando sus tablas. Los dispositivos que sí formen parte del grupo *multicast* responderán con la PDU *route reply* hacia el nodo que originó la búsqueda.

Mantenimiento de rutas Las rutas encontradas pueden volverse inválidas en cualquier momento por diversas razones. Cuando un nodo del camino no consigue (de forma reiterada) redirigir un paquete hacia el siguiente salto, debe reportarlo hacia el nodo origen del paquete mediante el envío hacia atrás de una PDU de control con el comando *network status*.

Puesto que el procedimiento es costoso, y el fallo puede ser temporal, no debe ponerse en marcha inmediatamente sino cuando el número de fallos es reiterado y supera un cierto umbral ($nwkcRepairThreshold$) y/o se extiende durante un tiempo.

***Source Routing* y registro de ruta** El nivel de red *ZigBee* soporta un mecanismo de *Source Routing*(o encaminamiento en destino) que permite especificar la ruta que va a seguir un paquete por la red, mediante un listado de saltos. También soporta un mecanismo que permite registar la ruta seguida por un paquete en su recorrido por la red.

3.2.5.5. Transmisiones *broadcast* y *multicast*

En las secciones anteriores se han mencionado estos mecanismos pero no se han dado detalles sobre su funcionamiento.

La transmisión broadcast a nivel de red tiene como objetivo enviar un mensaje a todos los nodos que la integran, y utiliza como destino la dirección especial 0xFFFF. Puesto que no todos los nodos tienen visibilidad directa, el mensaje se propaga por la red al ser reenviado a sus vecinos por los nodos *routers* o *coordinadores* que lo reciban (normalmente haciendo uso para ello del mecanismo *broadcast* de la capa MAC). Para evitar el reenvío de forma circular por la red, los dispositivos routers y coordinador deben mantener una tabla con las direcciones origen y el identificador de las PDUs broadcast retransmitidas recientemente. Al no ser factible que los nodos de la red confirmen la entrega del mensaje al nodo originario, para mejorar la fiabilidad del envío en la transmisión de los mensajes *broadcast* se hace uso de un mecanismo denominado confirmación pasiva (*passive acknowledgement*), según el cual tras retransmitir un mensaje broadcast los dispositivos comprueban si sus vecinos (almacenados en la tabla de vecindad) lo retransmiten, señal de que lo han recibido adecuadamente. Para evitar colisionar con otros dispositivos que estén difundiendo el mensaje por la red, y que no sean fácilmente detectados por el mecanismo CSMA/CA de la capa MAC debido al problema del nodo oculto, el mensaje no se reenvía inmediatamente tras ser recibido, sino que introduce un tiempo de espera aleatorio denominado *broascast jitter* cuyo valor máximo es un parámetro de configuración del nivel de red.

La transacción multicast permite el envío del mismo mensaje a un conjunto de dispositivos que forman parte de un grupo multicast identificado por una dirección de grupo de 16 bits y que están separados por un número de saltos inferior a $MaxNonMemberRadius$. Si el envío se origina en un dispositivo del grupo, se reenvía al resto mediante broadcast con un radio limitado (campo de la PDU que se decrementa en cada salto para limitar el número de veces que un paquete se reenvía). Esta transmisión *broadcast* puede ser propagada incluso por dispositivos que no son miembros del grupo, pero no se aplica el mecanismo de confirmación pasiva. Si el envío al grupo procede de un dispositivo que no pertenece al grupo, en primer lugar se hace una búsqueda de camino *multicast*, que crea un camino *unicast* hacia uno de los miembros del grupo. La transmisión se hace de forma *unicast* hacia éste, que luego lo retransmite (mediante broadcast) al resto de miembros del grupo.

3.2.5.6. Recepción en los dispositivos finales

Aquellos dispositivos, como los nodos finales, que utilizan transmisión indirecta para no tener la radio continuamente activada deben despertarse regularmente para extraer los datos del dispositivo padre. Este procedimiento es iniciado por la propia capa de red cuando determina que puede tener PDUs de control de nivel de red que recibir, o puede ser a su vez indicado a la capa de red por los protocolos de nivel superior.

El procedimiento para realizar la recepción es diferente según el modo de operación de la red. Si la red es no balizada, se ordena directamente al MAC realizar el sondeo del dispositivo padre mediante la primitiva *MLME-POLL.request*, mientras que si la red es

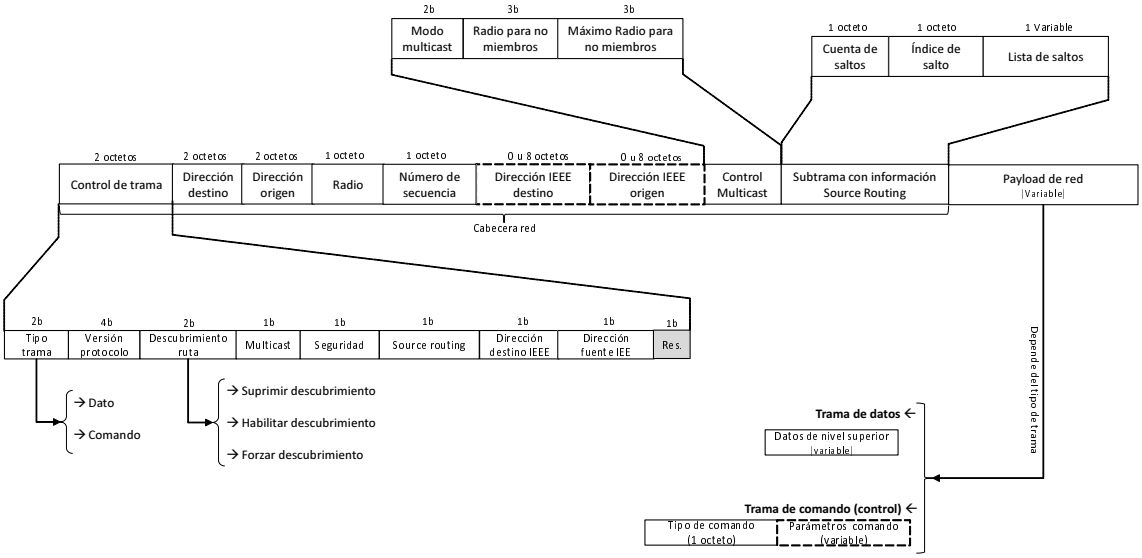


Figura 3.12: Formato general de las tramas o PDU (Protocol Data Unit) de nivel de red.

balizada, se solicita al MAC realizar la sincronización con el padre mediante la primitiva *MLME-POLL.request*, que hace que el dispositivo comience a seguir las balizas del padre. Mediante dichas balizas el MAC puede determinar si hay mensajes indirectos pendientes para él y en tal caso el propio MAC intenta automáticamente extraerlos del padre para pasarlos hacia el nivel de red cuando estén disponibles mediante la correspondiente primitiva de indicación del servicio de transporte de datos.

3.2.5.7. Formato de trama

El formato general de trama o PDU de capa de red se muestra en la figura 3.12. Hay dos tipos principales, las PDU de datos, que se utilizan para el transporte de datos de nivel superior, y las PDU de control que se utilizan para intercambiar comandos de control y configuración de la propia capa de red entre las entidades de gestión de red de diferentes dispositivos.

La trama está formada por:

- **Campo de control de trama.** Está compuesto por 16 bits que indican el tipo de trama, la versión del protocolo, y si se está empleando seguridad en la transmisión, principalmente. También hay otros bits de control que determinan si se debe hacer un registro de ruta o si se debe aplicar source-routing, si se trata de una comunicación multicast, etc.

- **Dirección corta de destino.** El valor 0xFFFF se usa para enviar el paquete a todos los nodos dentro del rango de alcance.

- **Dirección corta de origen.**

- **Radio.** Cada dispositivo que reciba la trama restará uno al valor de esta variable. Así podremos limitar el máximo número de saltos de la trama.

- **Número de secuencia.** Gracias a la dirección origen y al número de secuencia se puede identificar una trama de forma unívoca.
- **Dirección IEEE destino.** Su uso es opcional y es indicado en el campo de control de trama.
- **Dirección IEEE origen.** Su uso es opcional y es indicado en el campo de control de trama.
- **Campo de control Multicast.** Su uso es opcional y es indicado en el campo de control de trama. Define parámetros necesarios para la transmisión multicast.
- **Subtrama de ruta origen.** Su uso es opcional y es indicado en el campo de control de trama.
- **Carga útil o *payload***, que lleva la información de nivel superior en el caso de las PDU de datos, y la información de control o comandos a intercambiar por las entidades de gestión de la capa de red en el caso de las PDUs de control. En este último caso el *payload* está dividido en dos campos:
  - un campo identificador de comando que contiene un código que indica de qué comando se trata (*route request*, *route reply*, *leave*,etc...)
  - un campo *payload* de comando con campos que dependen del tipo de comando y que contienen información relacionada con el mismo (por ejemplo, en el comando *route request*, el identificador del comando, la dirección destino y el coste acumulado de la ruta)

3.2.6. Capa de aplicación *ZigBee*

La capa de aplicación es el nivel más alto de la pila de protocolos *ZigBee* y está compuesta por varios componentes: la subcapa de soporte de aplicación (APS), el marco de aplicación (application framework) y el objeto de dispositivo *ZigBee* (*ZigBee Device Object* o ZDO). Antes de entrar en más detalles de los diferentes componentes, resulta conveniente definir el significado de varios términos que aparecen de forma recurrente al describir la capa de aplicación:

- **Punto de acceso (endpoint)**. La capa de soporte de aplicación hace de interfaz entre las aplicaciones y la capa de red, permitiendo la multiplexación del transporte de datos de las diversas aplicaciones a través de ésta. Por cada objeto de aplicación, se crean y asignan diferentes puntos de acceso al servicio de transporte de datos, cada uno de los cuales tiene una dirección o identificador de *endpoint* (algo similar a lo que sería el concepto de puerto en los protocolos TCP/UDP). Dado que el identificador es de 8 bits, se podrían soportar hasta 256 puntos de acceso a servicios, aunque el punto de acceso 0 está asignado al nivel ZDO, el 255 es usado para la comunicación *broadcast* con todas las aplicaciones dentro del marco de aplicación, y el rango de 241-254 está reservado para futuros usos especiales.

- **Perfil de aplicación.** Es una especificación de mensajes, formatos, acciones y procedimientos que deben ser empleados por los objetos de aplicación para permitir la implementación de la aplicación y garantizar la interoperabilidad entre los dispositivos que participan en ella. Por ejemplo, el perfil de aplicación que define un sistema domótico es distinto al que define un sistema de control de sensores para la industria o al de un sistema de telemedida de contadores (*Smart Metering*). Estos perfiles permiten a las aplicaciones enviar comandos, intercambiar datos y procesar peticiones. Los perfiles son identificados mediante un valor de 16 bits asignado por la *ZigBee Alliance*. Cada perfil de aplicación está formado a su vez por un conjunto de clusters.

- **Cluster.** El cluster, identificado por un identificador ClusterID de 16 bits, permite indentificar la información que entra o sale del objeto de aplicación a través de un *endpoint*. El identificador de un cluster es único dentro del contexto de un perfil de aplicación. Los clusters pueden ser bien entrantes (se usan para recibir información) o salientes (se usan para enviar información). El cluster puede ser a su vez un contenedor de un conjunto de atributos que se utilizan en la comunicación de los distintos dispositivos ZigBee. Por ejemplo, dentro de un sistema de domótica, se puede definir un cluster para el control de luces, otro para el control de temperatura, etc.

- **Atributos de aplicación** Son una entidad de datos representando una magnitud física o un estado. Los atributos se utilizan en la implementación de servicios y pueden ser comunicados a otros dispositivos usando comandos. Por ejemplo, el estado de un interruptor de una bombilla (encendido/apagado) sería un atributo.

- **Vínculación (binding).** Es una conexión lógica entre un punto de acceso origen y uno o varios de destino. Sólo se puede realizar un vínculo entre puntos de acceso que compartan un mismo clusterId, siendo en uno saliente y en el otro entrante. Un ejemplo en una aplicación domótica sería un interuptor y una bombilla, donde el primero tendría un cluster ID saliente que sería compatible con el entrante de la bombilla. Puesto que un dispositivo puede poseer varios puntos de acceso, también puede soportar varios vínculos. La conexión lógica resultante es gestionada por la capa de soporta de aplicación, aunque se realiza a petición de los niveles superiores.

3.2.6.1. Capa de soporte de aplicación (APS)

La capa de soporte de aplicación realiza principalmente las funciones típicas de un nivel 4 OSI (nivel de transporte), ya que hace de interfaz con el nivel de red y se encarga de:

- Generar la PDU a nivel de aplicación. Una vez los puntos de acceso de dos dispositivos están vinculados, la capa APS es la encargada de gestionar el intercambio de mensajes.

- Gestión de la vinculación y envío de datos entre dispositivos vinculados.

- Proporcionar un transporte fiable de la información extremo a extremo.

- Fragmentación y reensamblado: En la variante *ZigBee-PRO*, permite el transporte extremo a extremo de mensajes con un tamaño superior al máximo *payload* de nivel de red.
- Filtrado de mensajes recibidos por duplicado.
- Gestión de grupos y de la seguridad.

Almacenamiento de datos persistente Además de lo anteriormente mencionado, la capa de soporte de aplicación es responsable de almacenar de forma persistente una serie de estructuras de datos que se utilizan tanto para su funcionamiento como para el de otras de las capas de nivel de aplicación. Entre ellos pueden mencionarse:

- La tabla de vinculación local (*binding table*, que permite guardar las entradas con los vínculos que el dispositivo tiene actualmente.
- Las tablas de descriptores. Permiten almacenar descriptores que permiten al entorno de aplicación (AF) realizar el descubrimiento de capacidades y servicios (entre ellos descriptores del nodo, de potencia y el descriptor simple de cada *end point*),
- La caché de tablas de vinculación, que permite a ciertos dispositivos almacenar la tabla de vinculaciones de otros.
- La caché de descubrimiento de servicios, que es utilizada por ciertos dispositivos, como por ejemplo el coordinador o algunos *routers*, para almacenar los descriptores de servicios de otros dispositivos.

Esta información debe intentar mantenerse incluso durante un fallo de alimentación, reset y otros eventos. Aunque es almacenada por la subcapa APS, en algunos casos es realmente utilizada por otros componentes de la capa de nivel de aplicación, como por ejemplo la caché de tablas de vinculación y la caché de descubrimiento de servicios, que son utilizadas por el ZDO. Muchas de estas estructuras sólo se encuentran en algunos tipos de dispositivo o en dispositivos concretos de la red

Vinculación y creación de grupos La capa APS mantiene una tabla de vinculación que permite a los dispositivos determinar los destinos a los que enviar las tramas proporcionadas a través de un determinado *end-point* y para un determinado clusterID. Cada destino puede representar un endpoint específico de un dispositivo específico o una dirección de grupo. La tabla de vinculación tendrá una entrada para cada pareja *end-point* origen y *clusterID*, y en cada entrada una lista de posibles destinos (parejas de direcciones y endpoints destino, o bien grupos de direcciones). Las entradas en la tabla de vinculación son añadidas o actualizadas por los niveles superiores, y utilizadas por el nivel APS al enviar los datos de nivel superior recibidos por los *endpoints*. Las direcciones de dispositivos utilizadas en las tablas de vinculación, son las direcciones IEEE de 64 bits, ya que las direcciones cortas de 16 bits, utilizadas para el transporte de datos por la red, son efímeras y pueden cambiar al cambiar la topología de red en una operación de mantenimiento.

Además de la tabla de vinculación, la capa APS también mantiene una tabla de grupos, que se utiliza para clasificar la información recibida utilizando el direccionamiento y entregarlas de forma selectiva a determinados *endpoints*. Es decir, la tabla de grupos permite asociar determinados end-points a direcciones de grupo. Las información sobre direcciones de grupo debe mantenerse consistente con la información sobre ellos que mantiene la capa de red.

Servicio de transporte de información Las capas de nivel superior pueden solicitar a la capa APS el envío de datos hacia otros dispositivos. Estos envíos se pueden hacer indicando la dirección (de 16 o 64 bits) y el *endpoint* del dispositivo destino (además del *profileId* y el *clusterId*) o bien indicando únicamente el *endpoint* origen. También se pueden realizar envíos indicando direcciones de grupo.

La capa APS es capaz de gestionar los diferentes tipos de envío, llevando cada uno a cabo de la forma más apropiada. Por ejemplo, si no se indica dirección destino, y sólo se indica el *endpoint* origen, el *profileId* y el *clusterId*, la capa APS consulta la tabla de vinculaciones y envía la información a todos los destinos que figuren en la correspondiente entrada (o descarta el envío e informa al nivel superior si no encuentra una entrada válida). Esto es válido tanto para destino que sean dispositivos como para grupos de direcciones.

Cuando el destino es una dirección IEEE de 64 bits proporcionada por el nivel superior o leída de una entrada de la tabla de vinculaciones, la dirección de 64 bits debe traducirse a la correspondiente dirección de 16 bits para poder enviar el los datos por la red. Para ello se hace uso de una caché de traducción de direcciones que es almacenada y gestionada por el nivel de red y que también puede ser manipulada por otros componentes como el ZDO.

El envío del paquete se realiza mediante el mecanismo más apropiado de la capa de red: *unicast* si se trata de un dispositivo concreto, *multicast* si se trata de un grupo o *broadcast* si se trata de un grupo pero el dispositivo no soporta *multicast*.

Los datos recibidos con una dirección de grupo serán entregados en aquellos *endpoints* locales que figuren en la correspondiente entrada de la tabla de grupos, o descartados si no figuran.

El envío de información se puede realizar con o sin confirmación extremo a extremo. En caso de enviarlo con confirmación, la capa APS del nodo destino deberá mandar de vuelta hacia la APS del origen un paquete de confirmación. Hasta que este no se reciba, no se confirmará al nivel superior la correcta finalización de la operación de envío. En este modo, la capa APS origen espera durante un tiempo $apscAckWaitDuration$ la llegada de la PDU de confirmación, realizando una retransmisión de la PDU de datos si no se recibe transcurrido dicho tiempo. La retransmisión se reintenta un número de veces $apscMaxFrameRetries$ (valor por defecto 3) antes de indicar al nivel superior el fallo de la operación. El valor por defecto de $apscAckWaitDuration$ se fija en función de la máxima profundidad del árbol, como el caso peor de recorrerlo en sentido ascendente y descendente y suponiendo 0,05 segundos de retardo de transmisión por salto.

En la variante *ZigBee-PRO*, la capa APS es capaz de dividir los paquetes de datos de nivel superior que lo requieran en varios fragmentos para su transporte por la red y

reensamblarlos en el destino. Esto mecanismo sólo es aplicable a paquetes *unicast* enviados con confirmación. Una vez fragmentados los datos en varios bloques, éstos son enviados en ventanas de hasta 8 bloques que deben ser confirmados en su totalidad antes de transmitir la siguiente ventana de bloques (el tamaño de la ventana, $apscMaxWindowSize$, es un parámetro de configuración). La capa APS del nodo destino debe enviar de vuelta paquetes de confirmación selectiva (ACK) para informar de la correcta recepción de todos los bloques de una ventana o para indicar aquellos que no se han recibido de forma correcta y que deben retransmitirse. En este escenario también se aplica el temporizador de retransmisión con valor $apscAckWaitDuration$ y que se reinicia, junto con la cuenta de retransmisiones cada vez que se recibe un paquete de confirmación que confirma algún bloque no confirmado anteriormente, aunque no confirme todos los paquetes que hay en la ventana. Para evitar mandar ráfagas de bloques hacia el nivel de red, éstos se envían con un tiempo de espaciado $apsInterframeDelay$ entre ellos.

La entidad APS del dispositivo destino debe acumular los bloques de transmisiones fragmentadas que vaya recibiendo, y también debe utilizar un temporizador para controlar el proceso. Si transcurre un lapso de tiempo superior a $aapscAckWaitDuration \cdot (apscMaxFrameRetries + 1)$ sin que lleguen nuevos bloques, la recepción se puede considerar fracasada y se pueden descartar los bloques recibidos. El receptor no genera paquetes de reconocimiento por cada bloque recibido, sino únicamente cuando se recibe el último bloque de la ventana (se hayan recibido o no los primeros) o cuando se recibe un bloque que precede a un conjunto de bloques de la ventana que ya se han confirmado previamente.

Formato de trama En la figura 3.13 se muestra el formato general de trama de capa de soporte aplicación, que está compuesta por:

- Campo de control de trama. Está formado por 8 bits que indican el tipo de trama, si se utiliza seguridad, si estamos empleando la extensión de cabecera y si se requiere confirmación ACK a nivel de aplicación, etc.
- Dirección del punto de acceso (*endpoint*) destino.
- Dirección de grupo. Si se usa direccionamiento de grupo la trama no debe incluir la dirección del punto de acceso destino. Todos los puntos de acceso asociados a la dirección de grupo en cuestión recibirán esta trama.
- Cluster ID. Identificador de cluster.
- Perfil ID. Identificador del perfil de aplicación.
- Dirección del punto de acceso (*endpoint*) origen.
- Contador APS. Se usa para evitar la recepción de tramas duplicadas.
- Extensión de cabecera. Extiende la funcionalidad de la cabecera en la versión *ZigBee-PRO* y permite básicamente el envío de información fragmentada y los reconocimientos selectivos.

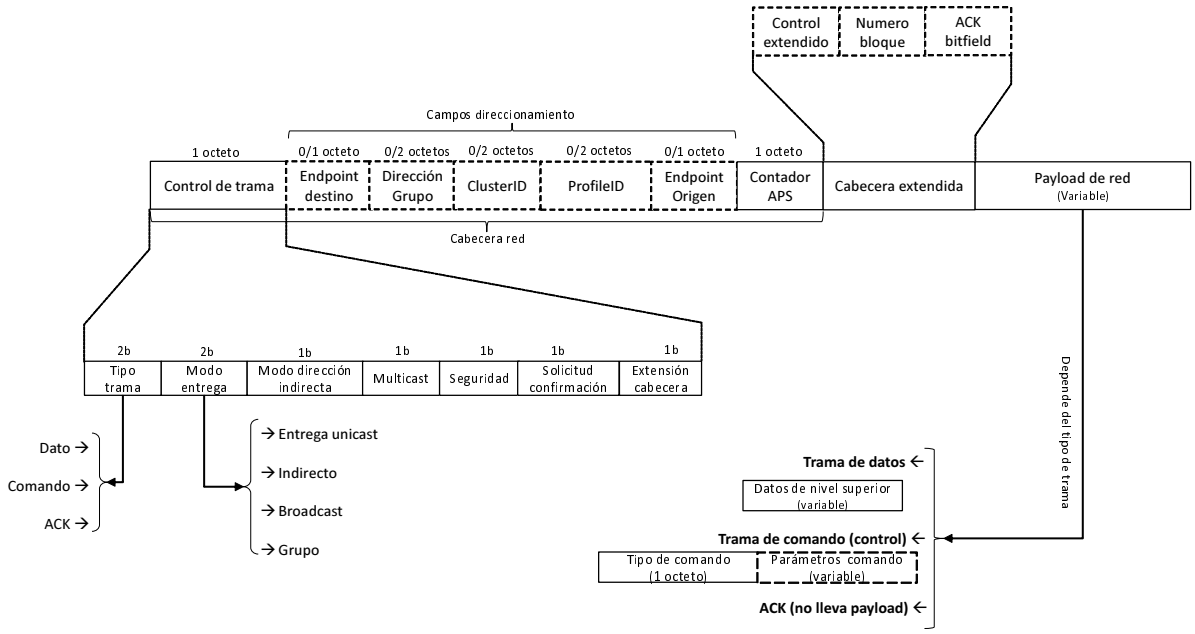


Figura 3.13: PDU de la capa de soporte de aplicación.

- Carga útil o *payload*, cuyo contenido depende del tipo de trama (si es datos, control o ACK). Al igual que en los niveles anteriores, las PDU de datos portan información de nivel superior, las PDU de control transportan comandos de gestión del nivel de aplicación y sus parámetros, y los ACK no llevan *payload*.

Marco de aplicación (AF o Application Framework) El marco de aplicación es el entorno en el cual se gestionan las diferentes aplicaciones, y establece unas directivas de cómo estructurar los servicios de aplicación para asegurar la interoperabilidad de productos de diferentes fabricantes dentro de una misma aplicación y facilitar el descubrimiento de servicios en la red.

El descubrimiento de servicios de red no es realizado por el *Application Framework*, sino que es responsabilidad del *ZigBee Device Object* (ZDO), pero la realización de dicho procedimiento requiere de la existencia de una serie de estructuras de datos que sirven para describir los servicios. El *Application Framework* estandariza en qué consisten y cómo deben crearse estas estructuras para describir y ofrecer un servicio dado.

3.2.7. *ZigBee Device Object*

Esta entidad del nivel de aplicación interacciona con las capas APS y de red para implementar dispositivos finales, routers y coordinadores *ZigBee*. A diferencia de otros objetos de aplicación, no sólo interacciona con la capa APS a través del *endpoint* 0, sino que también lo hace a través del punto de acceso al servicio de gestión de la capa APS.

El acceso al *endpoint* 0 permite a las entidad ZDO de un dispositivo interactuar con sus homólogas de otros dispositivos en la red para intercambiar información de control. Al igual que en el resto de objetos de aplicación, el ZDO se implementa siguiendo un

perfil definido por el estándar, y que se denomina *ZigBee Device Profile*. Este perfil tiene un único descriptor de dispositivo y emplea Clusters para definir el servicio. Es decir, los identificadores de Cluster (ClusterId) son utilizados dentro de este perfil para enumerar los mensajes que se pueden intercambiar entre los ZDO de diferentes dispositivos. No obstante, el perfil ZDP no utiliza atributos, por lo que la interacción entre ZDOs es muy similar a la aproximación basada en el intercambio de PDUs utilizadas por los protocolos de nivel inferior.

El *ZigBee device profile* proporciona soporte para realizar operaciones como el descubrimiento de dispositivos y servicios en la red, y para la gestión de la vinculación entre dispositivos (recuérdese que la capa APS almacenaba y utilizaba las tablas de vinculación, pero su actualización se hacía en base a peticiones del ZDO a través del SAP de gestión). El descubrimiento de dispositivo es el mecanismo para averiguar la identidad de otros dispositivos en la red y realizar correspondencias entre una dirección larga y una corta. Durante el descubrimiento de servicios el dispositivo local puede solicitar a otros dispositivos que proporcionen información detallada sobre su identificador de perfil, descriptores de servicios y lista de clusters de entrada y salida, lo cual puede ser utilizado para enlazar dispositivos durante el proceso de vinculación.

El ZDP permite que una entidad ZDO pueda operar (según el escenario) como cliente o como servidor. El cliente es el dispositivo ZDO que solicita un servicio como un descubrimiento de dispositivo o una vinculación, mientras que el ZDO del dispositivo que responde actúa como servidor. La especificación ZDP describe ambos comportamientos y por ello los servicios que ofrece se dividen entre servicios de cliente y servicios de servidor. En ambos casos los servicios son proporcionados mediante comandos representados por clústeres con un identificador único.

Las funciones principales del ZDO son:

- Inicializar y establecer la configuración de las capas de nivel inferior.
- Decubrimiento de dispositivos y nodos: permite solicitar información tal como la dirección de red y la lista de descriptores de cualquier otro dispositivo en la red. También permite al dispositivo registrar su información en la caché de caché de descubrimiento de servicios de otro nodo.
- Gestión de vínculos: permite a un dispositivo crear y eliminar vinculaciones, guardar la tabla de vinculación en la caché de tablas de vinculación de otro dispositivo, etc.
- Gestión de red: permite realizar operaciones relacionadas con la gestión de la red, como solicitar la tabla de enrutamiento y de vecinos de un dispositivo remoto, descubrimiento e identificación de redes vecinas, etc. Además se encarga de determinar y configurar el rol del dispositivo (coordinador, *router* o *end-device*)
- Gestión de nodos (opcional): Permite el control remoto de otro nodo, forzándolo remotamente a que se vincule, abandone la red, realice una búsqueda de redes.
- Gestión de grupos.
- Gestión de seguridad: Establecimiento e intercambio de claves, autenticación, etc.

3.2.8. Seguridad

Desde el punto de vista de la seguridad, las redes inalámbricas son altamente vulnerables debido a que no se requiere tener acceso a un medio cableado para participar en las comunicaciones. En los proprios estándares *ZigBee* y 802.15.4 se definen mecanismos de seguridad para las capas MAC, de red y de aplicación. Los mecanismos de seguridad deben asegurar tanto la confidencialidad de los datos como su autenticidad, sin suponer una carga computacional elevada para los dispositivos ni aumentar excesivamente la sobrecarga de las cabeceras.

En caso de que la seguridad esté habilitada, el coordinador es normalmente el que realiza las funciones de centro de seguridad, permitiendo o no la asociación de un nuevo dispositivo y generando y distribuyendo las claves que se usan en las comunicaciones.

3.3. Trabajos relacionados

ZigBee es una de las tecnologías inalámbricas comerciales más adecuadas para dar soporte a aplicaciones en las que se requiera la formación de redes con un número alto de dispositivos con restricciones de alimentación, y entre las que se encuentran particularmente las redes de sensores (WSNs) de bajo consumo. Al ser un estándar que define la totalidad de la pila de protocolo, tiene que dar respuesta a la mayor parte de los desafíos planteados tradicionalmente por éste tipo de redes y que han sido objeto de un creciente interés a lo largo de la última década.

El comportamiento de la capa MAC es uno de los elementos tradicionalmente considerados claves en el funcionamiento de las redes WSN, ya que determina en gran medida las prestaciones de la comunicación, la eficiencia en el uso del canal radio compartido y el rendimiento energético. En consecuencia muchas líneas de investigación se basan en la evaluación del funcionamiento del MAC especificado por 802.15.4 y la propuesta de mecanismos para mejorar la coordinación entre los nodos que participan en la comunicación. Por otra parte, el encaminamiento eficiente de la información para permitir la comunicación entre dispositivos que no tienen comunicación radio directa, y la evaluación de las prestaciones de la comunicación extremo a extremo, incluyendo la optimización de los diferentes mecanismos que aseguran que dicha comunicación es fiable, son otras de de las cuestiones más habitualmente abordadas por la comunidad investigadora.

Puesto que el soporte de dispositivos con restricciones de alimentación es uno de los objetivos inherentes a *ZigBee*/802.15.4, el consumo energético en habitualmente considerado en muchos de los estudios como una de las métrica de bondad utilizadas en la evaluación de la propia tecnología y de las configuraciones y mejoras propuestas, si bien el comportamiento energético en sí también es objeto de estudio específico en diversos trabajos.

3.3.1. 802.15.4

Muchos de los trabajos que estudian o intentan optimizar el rendimiento del mecanismo de control de acceso al medio de 802.15.4 están basados en simulaciones. Así por

ejemplo en [ZL04; ZL06] se lleva a cabo mediante NS-2 uno de los primeros estudios realizados sobre el funcionamiento del mecanismo de control de acceso al medio en diferentes modos, así como una comparativa con una red similar basada en 802.11. En [ZWDL08] se estudia el rendimiento del procedimiento de asociación a la red y se compara mediante simulación de eventos discretos con una propuesta alternativa realizada por sus autores. El trabajo [RGH+09] analiza (de nuevo con NS-2) el comportamiento del MAC para diferentes parámetros y escenarios con topología en estrella en presencia de nodos ocultos para diferentes cargas de tráfico y niveles de interferencia. Los autores de [JZLZ13] analizan mediante un modelo para NS-2 el efecto de la movilidad de los nodos en las prestaciones del protocolo, centrándose en particular en el estudio de la degradación de las prestaciones en escenarios de alta movilidad y constatando que en tales escenarios los mecanismos recogidos en el estándar no son eficientes. Algunas modificaciones para mejorar este comportamiento reduciendo la duración del procedimiento de *orphaning* se proponen en [TWP13] y se evalúan utilizando el entorno de simulación Castalia [Bou11] para OMNET++.

Otros trabajos combinan estudios mediante simulación con algunas pruebas reales. Por ejemplo, [SAB+07] realiza un estudio mediante simulación de las prestaciones de la capa física de 802.15.4 pero también investiga el efecto de posibles interferencias con otras tecnologías que operan en la misma banda (principalmente *Bluetooth* y *WiFi*) mediante pruebas con dispositivos comerciales. En otros estudios [PRML06; WW08b], se llevan a cabo ensayos en un entorno real para medir la tasa de error de paquetes (PER) y la potencia de señal recibida (RSSI) en función de la distancia entre los dispositivos, si bien posteriormente se evalúan las prestaciones de la capa MAC mediante simulación de eventos discretos con NS-2. En [PEFSV13] se propone un mecanismo para reducir el tiempo de actividad de los nodos en 802.15.4 en modo no balizado, y para evaluarlo se implementa sobre el sistema operativo Contiki para una red de nodos TelosB. El mecanismo propuesto sin embargo no es compatible con la especificación actual de la norma.

En [Lee06] se comparan mediante diferentes métricas (tasa de datos y tasa de pérdidas) las prestaciones de las transmisiones directas e indirectas en los modos balizado y no balizado de 802.15.4. El estudio se lleva a cabo en un escenario de pruebas real con un coordinador y un dispositivo final en presencia de otros tres dispositivos que introducen tráfico interferente aunque no participan en la red.

El comportamiento de los mecanismos de control de acceso al medio de 802.15.4, ha sido también modelado analíticamente en numerosos estudios, tanto para redes balizadas [TM06; PKC+05; SHC11; SS11; WGL14; PMSM15] como no balizadas [CLWY07; SS08; BV09; GRX+11]. La mayoría de estos estudios contemplan principalmente el modelado del algoritmo CSMA/CA ranurado o no ranurado que se emplea en las transmisiones directas, y comparan los resultados del análisis con simulaciones realizadas mediante eventos discretos para tráfico de las mismas características, pero en cambio obvian completamente el estudio del mecanismo de transmisión indirecta al que ni siquiera mencionan. Este modo sí es considerado específicamente en otros estudios analíticos [MSM06; PEE+08; Mis08a], aunque en el último se estudian las prestaciones de un hipotético dispositivo *bridge* que actúa como hijo de dos coordinadores distintos, algo en principio no previsto

por la especificación.

Otros trabajos han abordado también el estudio de las posibles interferencias entre 802.15.4 y otras tecnologías como *Bluetooth* y *WiFI* mediante un análisis matemático validado con simulaciones [SPCK07; SKP09; SPK07; FJYH12].

Aunque desde la aparición de las primeras versiones de 802.15.4 parece clara la necesidad de establecer una separación entre las supertramas de dispositivos vecinos para evitar que se produzcan colisiones, la especificación no recoge ningún mecanismo que permita gestionar la sincronización entre diferentes coordinadores para evitar solapes, como ya se mencionó en las sección 3.2.4.1. Las versiones más recientes de la norma *ZigBee* especifican claramente que se debe tener cuidado en intentar no solapar la supertrama de un dispositivo con las de sus vecinos, como se detalló en la sección 3.2.5.2, dando algunas indicaciones respecto a cómo localizarlas, pero dejando muchas cuestiones abiertas. Es por ello que numerosos trabajos de investigación han centrado su atención en la planificación de la duración y del instante de comienzo de las supertramas de los diferentes dispositivos con objeto de mejorar las prestaciones globales de la red al utilizar éste modo. El objetivo no es es sólo evitar colisiones entre dispositivos vecinos [CFLCG08; CHC09] sino también en muchos casos adaptar la longitud de la supertrama a las necesidades del tráfico [YP14; LYWA15] o mejorar el consumo [DAC+11]. Aunque por simplicidad la norma *ZigBee* restringe el uso del modo balizado a las redes en árbol, algunas investigaciones persiguen también extender su uso a redes parcialmente malladas [MPSP10]. El trabajo [KGM14] recoge una revisión bibliográfica de diversas propuestas realizadas en este sentido, junto con algunas otras para optimizar el comportamiento de la capa MAC, así como su interacción con las capas de nivel superior, al utilizar el modo balizado.

3.3.2. *ZigBee*

Muchos de los trabajos relativos a las prestaciones de *ZigBee* se centran en el funcionamiento del nivel de red y en la mayor parte de los casos presentan evaluaciones basadas en simulación. El trabajo presentado en [NS07] analiza las prestaciones de dos de los mecanismos de enrutamiento básicos de *ZigBee* y propone un nuevo algoritmo para mejorarlas. En [YCSK08] se utiliza también un entorno de simulación para comprobar el funcionamiento de algunos mecanismos que se propone añadir al comportamiento por defecto de *ZigBee* para facilitar un transporte fiable de datos en condiciones de congestión, como pueden ser aquellos escenarios en los que uno o pocos nodos son sumideros de información.

Algunos estudios utilizan también sistemas de pruebas reales. En [LWS12] se despliegan 51 nodos en un entorno *indoor* para realizar algunas pruebas básicas de pérdidas de paquete y *througput* máximo entre dos nodos. En [GYYL09] se realiza una prueba a menor escala con una red de cuatro dispositivos para comprobar el funcionamiento de una propuesta de arquitectura de red para el perfil *home automation* y comprobar la viabilidad de su integración con una red WiFi. [CTK10] se centra también en estudiar varios escenarios *indoor* constituidos por unos 10 nodos reales para evaluar los rangos de transmisión y la tasa de pérdidas de paquetes en función de la ubicación de los nodos en los escenarios, aunque no se estudia el impacto de los parámetros de configuración en las prestaciones.

Los autores del interesante trabajo descrito en [BGDP14] realizan un completo estudio

de diversos parámetros de *ZigBee* en un banco de pruebas considerablemente realista y formado por 60 nodos TelosB para los que desarrollan sobre TinyOS una implementación parcial de las diferentes capas de *ZigBee* que incluye principalmente los mecanismos relacionados con los parámetros de interés. Mediante dicho entorno de pruebas y configurando diferentes topologías, se comprueba el efecto de diversos parámetros de configuración como el número de reintentos de transmisión a nivel MAC, el mecanismo utilizado para estimar la bondad de los caminos en el nivel de red y el *timeout* de retransmisión extremo a extremo a nivel de aplicación, considerando el retardo de transmisión, la tasa de pérdidas y el gasto energético estimado como métricas para realizar la evaluación.

La implementación de mecanismos de enrutamiento alternativos para optimizar el comportamiento en algunos escenarios y aumentar el rendimiento es otra de las líneas de investigación que más interés ha atraído [Sar13]. Algunas propuestas se centran en la mejora del algoritmo de enrutamiento jerárquico en árbol aprovechando la información disponible sobre los nodos vecinos [CLMM07; SHK12; TSJ+14], mientras que en otras se proponen métricas y optimizaciones para el protocolo AODV [BK08; SZR+09; OROOB11]. Otros trabajos proponen también la sustitución de AODV por diferentes alternativas [HL10a; DR13]. Algunos estudios proponen mejoras específicas del mecanismo de enrutamiento para redes balizadas [HKPZ12; HPH+12] para tener en cuenta su interacción con las supertramas.

3.3.3. Consumo

El estudio [LSS07] recoge los datos de consumo en transmisión y recepción especificados en las hojas de características de *chipsets* comerciales para diversos estándares de comunicaciones inalámbricas (incluyendo *UWB*, *WiFi*, *Bluetooth* y 802.15.4/*ZigBee*), y realiza una comparativa básica entre ellos sin tener en cuenta las particularidades de sus diferentes modos de operación.

En uno de los estudios analíticos mencionados en la sección 3.3.1 [PEE+08], se propone además un modelo que permite predecir el consumo de energía por bit recibido mediante la estimación de la cantidad de tiempo que el transceptor radio permanece en cada estado (*Idle*, transmisión, recepción, *CCA*), en función de la carga de tráfico, el tamaño de paquete, el número de nodos y la configuración de parámetros del control de acceso al medio. En este caso el modelo se aplica únicamente a la transmisión directa.

Estudios parecidos se llevan a cabo en [KKHH06; WGL12] para un escenario *cluster-tree*, planteando una formulación matemática para calcular el consumo del coordinador y de los dispositivos finales dependiendo del tráfico emitido y de la temporización del modo balizado. En ambos trabajos los valores de consumo utilizados en el estudio son extraídos de la hoja de características de un transceptor comercial, el CC2420 de Texas Instruments, y el modelo se valida mediante simulación. Un procedimiento similar, aunque en este caso exclusivamente mediante simulación, es utilizado por los autores de [YPGS07] en la evaluación de su propuesta para optimizar el consumo de energía de los dispositivos mediante un mejor ajuste del control de congestión del mecanismo de contienda del CSMA/CA ranurado. El modelo de consumo utilizado en este caso desprecia el consumo de energía que tiene lugar en algunos estados como por ejemplo durante el cambio de sentido del

transceptor radio (*turnarround*) o durante los periodos de *backoff*.

El estudio del consumo en redes balizadas es también abordado por los autores de [BCD+08], que realizan sus propias medidas tanto para caracterizar el consumo de un transceptor comercial en diferentes estados como para determinar empíricamente la relación entre la potencia de señal recibida y la tasa de error de bit. Mediante las medidas obtenidas se desarrolla un modelo que tiene en cuenta la dinámica del algoritmo CSMA/-CA y permite obtener la energía media por bit transmitido en función de las pérdidas de propagación. Para ello es necesario estimar algunas estadísticas del funcionamiento de CSMA/CA (duración media del periodo de contienda, probabilidad de colisión, etc.) que son obtenidas mediante simulaciones por el método *Monte-Carlo* para un escenario y un tráfico concretos.

Los autores de [ZZZ09] plantean utilizar una estimación del estado de la batería como métrica para establecer la bondad de un camino al emplear el protocolo basado en AODV que establece el estándar de *ZigBee* para la búsqueda de rutas en redes malladas. El mecanismo propuesto es evaluado mediante simulación.

En [Mis08b] se desarrolla un modelo muy teórico para estudiar el consumo debido a la necesidad de transportar periódicamente claves por la red introducida por los mecanismos criptográficos empleados en *ZigBee*, y tras un análisis matemático de gran complejidad se derivan unos criterios para establecer la estructura de la red en *clusters* en función de la expectativa de vida de la misma. El efecto de los procedimientos de *orphaning* y reasociación a la red no son tenidos en cuenta en el estudio.

El trabajo [CWG+10] implementa un modelo de 802.15.4 para el *framework* INET de OMNET++, incluyendo también estimaciones del consumo basadas en el modelo de consumo propuesto en [LWG05] para el dispositivo MICA2, basado en un transceptor radio para la banda de 868 Mhz que en principio no es compatible con la especificación 802.15.4 para dicha banda.

El consumo de dichos procedimientos y de otros como la vinculación entre dispositivos es estudiado en [MKSM11] sobre módulos comerciales basados en el *chipset* AT86RF230 de ATMEL. Sin embargo en este estudio no se aislan los dispositivos de otras posibles fuentes de interferencia ni se tiene en cuenta el efecto de éstas en el consumo. Un estudio similar [Ami11] realizado mediante medidas sobre un dispositivo Tmote Sky con el *chipset* CC2420 de Texas Instruments propone un modelo lineal simplificado para estimar las cotas superior e inferior del tiempo de vida (duración de la batería) en una red de sensores inalámbricos. A pesar de que el estudio persigue predecir el tiempo de operación más largo que puede alcanzarse, ni las medidas realizadas ni el modelo contemplan la energía adicional necesaria debida a los intentos fallidos de acceder al canal ni a los mensajes perdidos debido a colisiones.

Algunos estudios de consumo se centran en determinar la adecuación de las tecnologías 802.15.4 y *ZigBee* a determinadas aplicaciones, como por ejemplo la creación de redes de área personal y corporal para aplicaciones médicas [TS04; GCR05; FLMA+09]. En concreto [TS04] presenta un modelo analítico para calcular el tiempo de vida de una hipotética red de sensores 802.15.4 implantados. En el estudio se utilizan los valores de consumo típicos del transceptor CC2420 y se lleva a cabo tanto para el modo balizado como

para el no balizado, y se concluye que el modo balizado presenta importantes restricciones en la tasa de datos y en la deriva temporal de los relojes de los dispositivos. Por otra parte [GCR05; FLMA+09] investigan la aplicabilidad de 802.15.4 a redes de sensores en el campo de la medicina mediante simulaciones sistemáticas realizadas con OPNET y OMNET++. Con objeto de estimar la energía consumida por mensaje transmitido se utilizan los datos especificados en la hoja de características del módulo JN5139 de Jennic.

El estudio [ACDN10] analiza la fiabilidad de los *cluster-trees* 802.15.4 frente a tres configuraciones de parámetros CSMA/CA diferentes. En base a este estudio, los mismos autores proponen en [DAC+11] un mecanismo *cross-layer* para ajustar los parámetros del MAC adaptativamente de forma que se minimice el gasto de batería, y lo evalúan mediante simulaciones con NS-2 tanto en redes con un único salto como para redes multisalto. [TMPS11] evalúa, mediante el entorno de simulación Castalia para OMNET++, el consumo de energía de los nodos en redes multisalto balizadas y no balizadas al utilizar diferentes estimadores físicos y lógicos de la calidad del canal. Estos tres estudios emplean también los datos de consumo del transceptor CC2420 de *Texas Instruments* anteriormente mencionado. Los autores de los dos primeros trabajos analizan de nuevo en [ACD11] la parametrización del MAC, en este caso utilizando un entorno de pruebas real basado en los módulos JN5139 de Jennic. En este estudio se mide una tasa de pérdidas de paquete del 0 a 5 %, aunque las condiciones en las que se inducen éstas pérdidas no se mantienen bajo control. Por otra parte en dicho estudio no se considera el consumo, ya que el objetivo es principalmente evaluar la fracción de paquetes entregados para compararla con los estudios por simulación.

Para estimar la duración de la batería habitualmente se considera un modelo simplificado en el que ésta tiene una capacidad nominal, que se va agotando conforme transcurre el tiempo debido a la corriente consumida, de ahí que el consumo medio de corriente sea caracterizado o considerado como métrica en la mayor parte de los estudios. Si bien es cierto que el consumo medio es el principal factor determinante en la duración de la carga de la batería, el comportamiento de una batería real depende también de su tecnología y puede verse afectado por otros factores (grandes corrientes de pico, autodescarga, etc.). Los autores de [FVV12] señalan algunos de estos factores y, con objeto de estudiar el tiempo de vida de diversos tipos de batería, implementan una plataforma que permite emular posibles patrones de consumo de un nodo con tecnología 802.15.4 bajo diferentes configuraciones.

3.4. Aportaciones al estudio y modelado del consumo en redes *802.14.4/ZigBee*

En esta sección se incluyen las aportaciones realizadas dentro del marco de prestaciones de consumo en redes *802.14.4/ZigBee*, y que se han recogido en diversos artículos publicados en congresos y revistas científicas[CCGCG10a; CCGCG10b; CCG10; CCG11; CCA12].

3.4.1. Sistema de prueba

Los escenarios de prueba utilizados en los estudios recogidos en esta sección se basan, con pequeñas variaciones, en el sistema esquematizado en la figura 3.14. La configuración básica de la red sigue una topología en estrella con un nodo coordinador, que actúa como sumidero de información, y un nodo *end-device* que actúa como nodo sensor. Aunque la topología utilizada es en estrella, la red utilizada soporta todos los niveles de la pila de protocolo *ZigBee* en su configuración mallada, por lo que no se utiliza el modo *balizado*. La caracterización del consumo se lleva a cabo para el nodo *end-device*, que es el que puede permanecer inactivo la mayor parte del tiempo gracias a la utilización de las transmisiones indirectas en sentido descendente (desde el coordinador hacia el *end-device*).

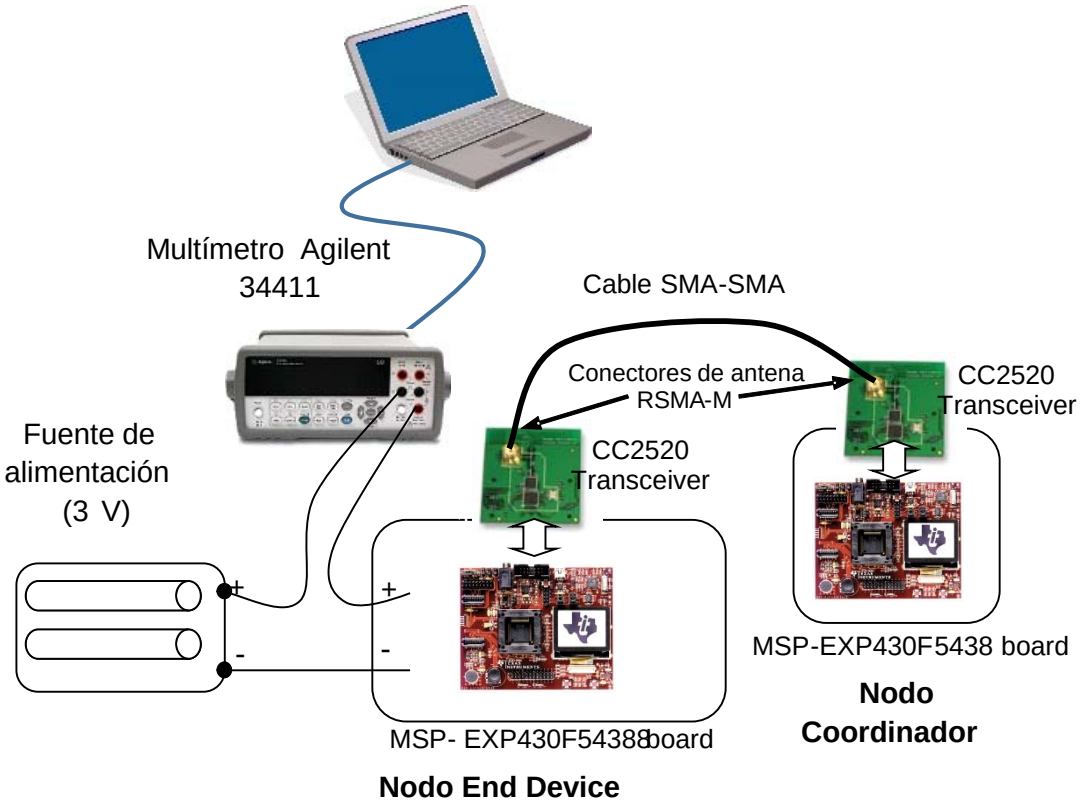


Figura 3.14: Sistema de pruebas *ZigBee/802.15.4*.

A lo largo del desarrollo de los estudios presentados en esta sección se han utilizado distintas plataformas *hardware* para caracterizar el consumo, la mayor parte basadas en sistemas de desarrollo de Texas Instruments, que es uno de los principales desarrolladores de *chipsets* compatibles con los diferentes estándares relacionados con 802.15.4. En concreto se han utilizado sistemas basados en el *ZigBee Processor* CC2480 y sistemas basados en el transceptor CC2520 controlado por un microcontrolador externo.

El transceptor CC2520 [Cc2a] opera en la banda de 2,4 GHz e implementa la capa física de 802.15.4 y algunas funcionalidades correspondientes al nivel MAC tales como la realización del CCA, el filtrado de tramas y la generación y envío automático de paquetes ACK, que requieren una temporización bastante estricta. Este transceptor es controlado

a través de un interfaz serie SPI por un microcontrolador externo en el que se ejecuta el resto de la pila de protocolos *802.15.4/ZigBee*. El microcontrolador presente en los sistemas de desarrollo utilizados con este transceptor es el MSP430F5438, que es una de las variantes con mayores prestaciones dentro de la conocida arquitectura de microcontroladores de bajo consumo MSP430de Texas Instruments y cuenta con 256 kB de Flash y 16 kB de RAM. El propio Texas Instruments proporciona una implementación de la pila completa de ZigBee/802.15.4 para esta plataforma, denominada *Z-Stack* [Zst], que es una de las soluciones comerciales más ampliamente utilizadas en la implementación de redes de sensores inalámbricos basados en esta tecnología.

La *Z-Stack* forma parte del *firmware* que se ejecuta en el microcontrolador, y aunque parte de la implementación es suministrada como bibliotecas ya compiladas, algunos componentes son suministrados como módulos de código en C, permitiendo modificar su comportamiento y/o su configuración. En algunos de los resultados incluidos en esta sección se ha estudiado el efecto en el consumo energético de las pérdidas de paquetes debidas a colisión y de los fallos de CCA. Para poder controlar exactamente su probabilidad, éstas pérdidas son en realidad emuladas modificando y recompilando el código fuente disponible de la *Z-Stack* antes de descargarlo al microcontrolador. En concreto, se alteran los procedimientos que realizan el CCA en el CSMA/CA no ranurado y la función que informa a los niveles superiores del protocolo de la correcta recepción de un ACK que reconoce un paquete enviado previamente. De esta manera, se fuerza la repetición del procedimiento de CSMA/CA con cierta probabilidad constante configurable en cada experimento en lugar de depender del estado del canal (el cual por otra parte debe ser siempre libre al estar conectados los dispositivos directamente mediante un cable coaxial en lugar de utilizar antenas). De la misma forma, en cada experimento se puede fijar una probabilidad de pérdida de paquetes por colisión. Esta en realidad se simula descartando los ACKs recibidos, como si éstos no hubiesen sido enviados por el otro extremo, lo que provoca la retransmisión del paquete original al cabo de un cierto tiempo. En ambos casos el procedimiento se implementa generando un número pseudoaleatorio comprendido entre 0 y 65535, que se normaliza a 1 y se compara con la probabilidad de fallo de CCA o de colisión establecida para determinar si la operación debe ejecutarse normalmente o se debe emular un fallo.

En el caso del CC2480, se trata de un sistema en chip (SoC) que integra una CPU con arquitectura 8051 junto con un transceptor 802.15.4 e implementa toda la funcionalidad *ZigBee*. La operación de este dispositivo es controlada mediante el envío de órdenes por un interfaz serie desde un procesador externo, y su *firmware*, que implementa la pila de protocolo *ZigBee* completa, se encuentra almacenado en la ROM del dispositivo y no es modificable. El sistema de desarrollo basado en este dispositivo también dispone de un microcontrolador externo para dar soporte a la aplicación y que es el encargado de poner en marcha y ordenar la realización de operaciones (como la conexión a la red o el envío de datos) al CC2480.

Además de los sistemas de desarrollo de Texas Instruments, también se han realizado en algunos casos medidas complementarias utilizando otras plataformas comerciales, como el MC1322x de Freescale. El MC1322x es un SoC programable que integra el transceptor

radio y una CPU con arquitectura ARM7. En este caso el *firmware* del dispositivo es parcialmente reconfigurable e incluye la implementación de Freescale de la pila de protocolo *ZigBee*, denominada comercialmente *BeeStack*.

Como ya se ha mencionado, para anular en la medida de lo posible el efecto de posibles interferencias de otras comunicaciones en la misma banda, los nodos transmisor y receptor están unidos directamente mediante un cable coaxial de 0,5 metros de longitud con conectores SMA, en lugar de utilizar antenas. Las pérdidas de potencia de transmisión debidas a los conectores SMA y al propio cable (de unos 2dB) son además así mucho menores a las que se producirían en la propagación por el espacio libre. El nivel de señal del transmisor se ha configurado a 0dBm, de forma que el nivel de señal que llega al receptor está muy por encima de su sensibilidad (-98 dBm) y por debajo del límite de saturación indicado en sus especificaciones (6 dBm). Esto prácticamente garantiza que todos los fallos de CCA y colisiones detectados vienen forzados por los mecanismos de emulación introducidos en la pila de protocolo del nodo *end-device*.

Para estimar la corriente media consumida por el nodo *end-device*, se utiliza un multímetro 34411 de Agilent (actualmente Keysight). Dicho multímetro puede ser controlado desde un PC a través de un interfaz USB, ofreciendo un entorno muy versátil que soporta diferentes configuraciones de medida. Para capturar la forma de onda (o variación rápida con el tiempo) de la corriente debida a las diferentes operaciones del dispositivo, se ha configurado el multímetro para realizar medidas con una frecuencia de muestreo de hasta 50 kS/s y 4,5 dígitos de resolución, mientras que para estimar el consumo medio de corriente se ha configurado el multímetro con la máxima resolución (6,5 dígitos) y tiempos de integración largos (2 segundos). El multímetro 34411 incorpora un conversor A/D de integración continua [Goe92; Blu92; Fuh02] que muestrea la señal integrándola durante todo el intervalo de muestreo sin introducir pausas para realizar la conversión, adquiriendo por tanto cada muestra como el valor medio de la señal en dicho intervalo. El valor medio de consumo en un intervalo de tiempo mayor se puede estimar como la media aritmética de las diferentes muestras adquiridas en dicho intervalo. Utilizando este último modo es posible realizar mediciones del consumo medio para diferentes configuraciones del dispositivo sensor con muy buena resolución. Para la realización automática de ambos tipos de medida se ha desarrollado una aplicación para el entorno de programación LabView que permite configurar y controlar la operación del multímetro y leer los datos adquiridos desde el PC.

Los sistemas de desarrollo utilizados son alimentados por dos baterías AAA de 1,5 V de tensión nominal, aunque en los experimentos realizados se ha utilizado en su lugar una fuente de alimentación de laboratorio. El *firmware* de los nodos caracterizados han sido configurado cuidadosamente de forma que se desactivan los diferentes periféricos de la placa y todas las funcionalidades que no son estrictamente utilizadas en la comunicación *ZigBee*. Igualmente, los pines de entrada salida GPIO del microcontrolador que no se utilizan se han configurados como salidas para evitar consumos espúreos debido a entradas sin conectar.

3.4.2. Caracterización del consumo de dispositivos *802.15.4/ZigBee*

En esta sección se presenta un conjunto de medidas de la corriente consumida por un dispositivo *end-device* al ejecutar algunas operaciones típicas que realizaría un nodo sensor para conectarse y enviar datos a la red 802.15.4/*ZigBee*

3.4.2.1. Consumo durante el arranque el dispositivo

La figura 3.15 muestra la corriente instantánea consumida por el dispositivo basado en el CC2480 tras su encendido. Inicialmente sólo se activa el microcontrolador presente en la placa de desarrollo (intervalo 1 marcado en la figura), el cual transcurrido aproximadamente un segundo desde el arranque y una vez comprobado que el voltaje de alimentación es estable, procede a aumentar su frecuencia de reloj, pasando a consumir unos 4,6mA. Tras este ajuste, el microcontrolador activa el procesador *ZigBee* CC2480, aumentando el gasto de corriente hasta 15,4mA (intervalo 3). A partir de ahí el dispositivo permanece en dicho estado hasta que se configura su modo de funcionamiento (haciendo uso de un botón presente en la placa de desarrollo).

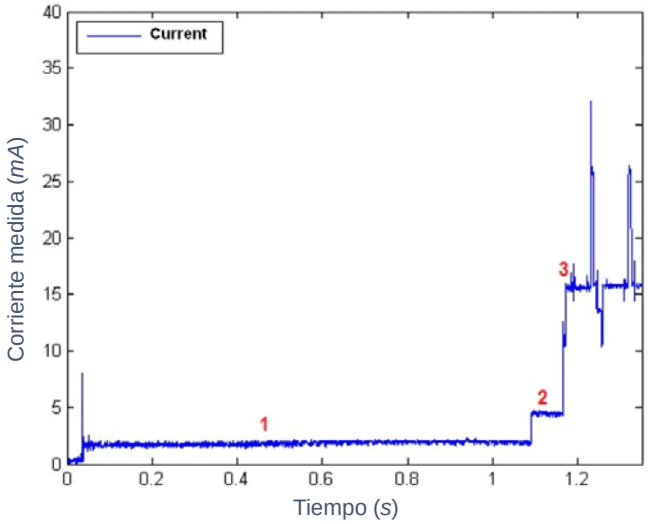


Figura 3.15: Corriente consumida durante el arranque y configuración inicial del dispositivo basado en el CC2480.

3.4.2.2. Consumo durante la asociación al coordinador

Para agregarse a la red, el dispositivo debe detectar la presencia de un nodo coordinador 802.15.4 (*router* o coordinador *ZigBee*) y realizar el procedimiento de asociación. El consumo de corriente requerido durante esta operación para el dispositivo basado en el CC2480 se ilustra en la figura 3.16, y se compone de las siguientes fases:

- Intervalos 1 a 3: Al finalizar la fase de arranque, el dispositivo queda a la espera de ser configurado por el usuario mediante la pulsación de un botón. Si se pulsa el botón dos veces durante un periodo de unos 2 o 3 segundos (que el caso de representado

en la figura), el dispositivo se configura en modo *end-device* (de lo contrario pasará comportarse como coordinador o router). El consumo durante esta fase sigue siendo similar al que tenía en la fase final del proceso de arranque (unos 15,4mA), puesto que tanto el procesador *ZigBee* CC2480 como el microcontrolador están activados.

- Intervalo 4: Una vez configurado el modo, el microcontrolador ordena al procesador *ZigBee* CC2480 iniciar la búsqueda de un nodo coordinador al que asociarse, tras lo cual el microcontrolador pasa a un modo de bajo consumo.

- Intervalos 5 a 7: En una red *ZigBee* mallada, los dispositivos utilizan el modo no balizado de 802.15.4, y por tanto el coordinador no indica su presencia a otros dispositivos mediante la emisión de tramas piloto. En consecuencia, el nodo sensor debe ejecutar el procedimiento de escaneo activo de canal para detectar la presencia de nodos coordinadores. Este procedimiento, que se describió previamente en la sección 3.2.4.5, se lleva a cabo para un conjunto de canales preconfigurado y permite obtener una lista de dispositivos coordinadores en la vecindad. En el caso del experimento correspondiente a la figura 3.16, se exploran en particular dos canales, correspondiendo con las fases 5 y 7 respectivamente. El pico marcado por el punto 6 de la figura corresponde al momento en el que se realiza la conmutación entre ambos canales. El proceso es posteriormente repetido otras dos veces, como puede observarse en la figura. El escaneo de cada canal tiene una duración aproximada de unos 0,5 segundos y el consumo medio durante el mismo es de unos 32,5mA debido principalmente a que la radio debe permanecer en modo recepción durante la mayor parte del mismo para recibir las posibles respuestas de los coordinadores vecinos.

- Intervalo 8: Tras la fase de escaneo y tras haber seleccionado un coordinador, se inicia la dase de asociación, también descrita previamente en la sección 3.2.4.5. Durante esta fase los dispositivos coordinador y *end-device* se intercambian una serie de paquetes para informar al *end-device* sobre la configuración de la red y arrancar su operación (por ejemplo se le comunica la dirección corta asignada). Tras la asociación a la red, también se realiza una vinculación a nivel de aplicación. La vinculación permite establecer un enlace lógico entre las aplicaciones en ambos extremos. En el caso del experimento mostrado, la vinculación requiere el envío de 7 paquetes entre el coordinador y el *end-device*. Estos dos procedimientos en su conjunto dura aproximadamente 1 segundo y el gasto de corriente medio es sensiblemente inferior que durante la etapa de escaneo.

- Intervalo 9: Finalmente, tras la asociación y vinculación, el dispositivo CC2480 entra en modo de bajo consumo, estado en el que permanece la mayor parte del tiempo excepto cuando sea necesario realizar alguna otra operación.

En la figura 3.17 se representa el consumo de corriente de los mismos procedimientos para el sistema de desarrollo basado en el transceptor CC2520 y el microcontrolador MSP430F5438, en el que se dispone de diversos puntos de test independientes que permiten medir la alimentación del microcontrolador (mostrada en color verde) y la del transceptor

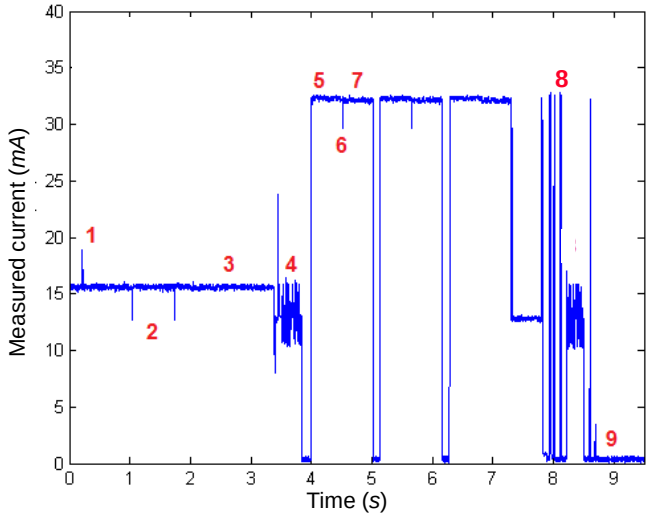


Figura 3.16: Corriente consumida durante la asociación y vinculación del sistema basado en el CC2480 al coordinador.

(mostrada en azul) por separado. En este caso en la subfigura 3.17.a se muestra el consumo durante el escaneo del dispositivo (un único canal), que se repite hasta encontrar un dispositivo coordinador. Cuando se encuentra un coordinador válido, se produce la asociación, cuyo proceso se muestra en la figura 3.17.b. Para este sistema la vinculación (subfigura 3.17.c) no se realiza hasta que el usuario pulsa un botón.

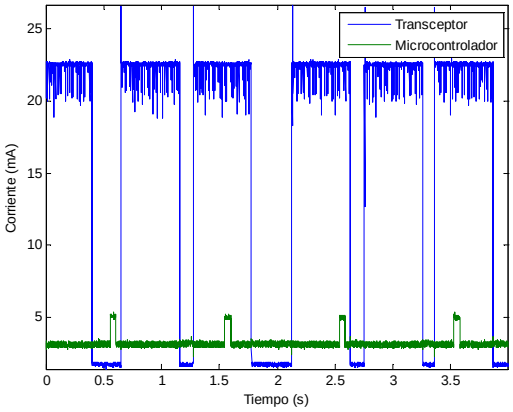
3.4.2.3. Consumo durante la transmisión de un paquete

La evolución temporal de la corriente requerida para transmitir un paquete de datos en el sistema de desarrollo basado en el CC2580 hacia el coordinador se representa en la figura 3.18.

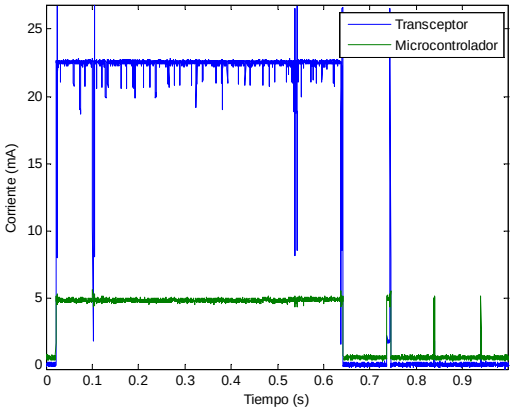
El proceso consta de varias fases que se detallan a continuación:

- Intervalos 1, 2 y 3: Durante este periodo, de unos 10,2 ms, se activa el CC2480 y su CPU, aunque no la radio. La corriente demandada es aproximadamente 13mA. El aumento de corriente que se observa en el punto 2 se debe a la actividad del microcontrolador MSP430 externo durante dicho intervalo.

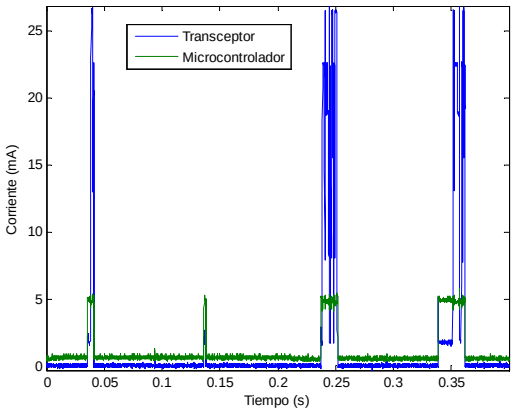
- Intervalo 4: El transceptor radio se activa y la corriente aumenta hasta 32.5 mA mientras el dispositivo comprueba si el canal está libre (CCA). El *firmware* de estos dispositivos, que no es modificable, permanece en este estado durante todo el *backoff*, aunque la norma no obliga a ello. El tiempo requerido para acceder al canal mediante la aplicación del algoritmo CSMA/CA depende obviamente de la ocupación del mismo. Puesto que en el sistema de pruebas utilizado en este experimento no se producen interferencias de otras comunicaciones, el canal está siempre libre, y por tanto el valor medio estimado para la duración de esta fase en diferentes iteraciones del experimento (aproximadamente 1,6ms) puede ser considerada como el tiempo medio de espera para el caso ideal.



a) Escaneo



b) Asociación



c) Vinculación

Figura 3.17: Corriente consumida durante la asociación y vinculación del sistema basado en el CC2520 al coordinador.

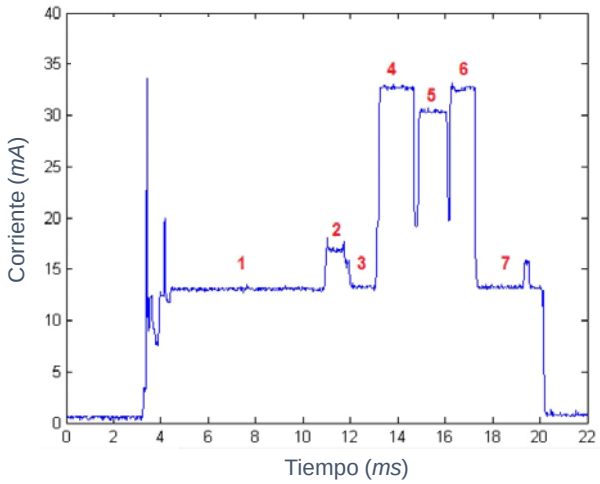


Figura 3.18: Corriente consumida durante la transmisión de un paquete de datos desde el nodo *end-device* basado en el CC2480 hacia el coordinador.

- Intervalo 5: Cuando finalizan los periodos de *backoff* que deben esperarse y una vez que se determina que el canal está libre, el dispositivo envía al coordinador el paquete de datos. La corriente necesaria para ello es de unos 30,5mA y el tiempo empleado depende de la longitud del paquete enviado (puede calcularse en función de ésta y de la tasa binaria).

- Intervalo 6: En esta fase, que dura aproximadamente 1,3ms, el dispositivo conmuta de nuevo la radio al modo recepción para esperar a recibir el paquete ACK que confirma la correcta recepción de los datos. La transmisión de dicho paquete, de 11-bytes de longitud, no requiere la aplicación del algoritmo CSMA/CA puesto que 802. 15.4 establece un tiempo de guarda tras la transmisión de un paquete en el que ningún nodo puede transmitir excepto para reconocer un paquete recibido. Durante este intervalo, el consumo vuelve a ser de is 32.5 mA, y tras él la radio se desactiva.

- Intervalo 7: Durante este tiempo, de unos 3 ms, se mantiene activa la CPU del CC2480 y se envía información de control hacia el microcontrolador externo, el cual se activa brevemente, y una vez finalizado el proceso el CC2480 vuelve al modo de bajo consumo.

La figura 3.19 muestra también el consumo durante la transmisión de un paquete en el dispositivo basado en el transceptor CC2520. En este caso, de nuevo es posible medir y representar por separado la corriente consumida por el transceptor y la cosumida por el microcontrolador. Con respecto al consumo del microcontrolador, hay que recordar que en esta implementación, a diferencia de la basada en el CC2480, el transceptor sólo implementa la capa física y algunos procedimientos de la capa MAC, siendo el *firmware* del microcontrolador el encargado de llevar a cabo el resto de comportamientos de la pila de protocolo 802.15.4/*ZigBee*. En este caso, en las fases 2, 3 y 4 se realiza la comprobación de que el canal está libre, la transmisión y el intento de recibir el ACK respectivamente. Durante la fase 1, el microcontrolador envía al transceptor la información necesaria para proceder al envío del paquete. Al igual que en el caso anterior, la duración de la fase 3 depende del tamaño del paquete y la de la fase 2 es aleatoria y depende de la ocupación del canal, siendo igualmente la duración media la que corresponde al caso ideal (no hay posibilidad de que el canal esté ocupado en las pruebas realizadas).

3.4.2.4. Consumo debido a la pérdida de conexión

Si un dispositivo *end-device* no recibe confirmación del coordinador tras reenviar los datos de forma repetida, la pila de protocolos *ZigBee* puede asumir que el dispositivo ha perdido la conexión (ha quedado «huérfano») y en tal caso intentar realizar una reasociación con el dispositivo coordinador. En este caso, el dispositivo «huérfano» realiza un escaneo activo de canal en los diferentes canales configurados, intentando volver a encontrar un coordinador de la red con la que estaba conectado. Este procedimiento se describió en detalle en la sección 3.2.4.5.

En lo que respecta al consumo de corriente debido a una pérdida de conexión, la corriente media empleada depende del número de canales escaneados. La figura 3.20 muestra

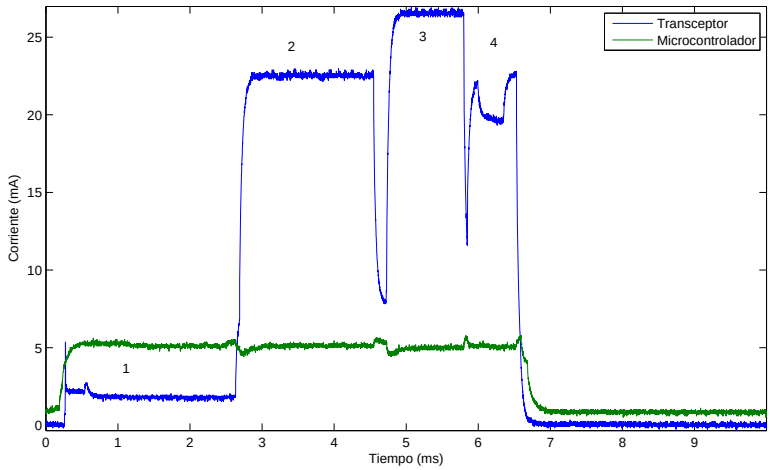


Figura 3.19: Corriente consumida durante la transmisión de un paquete de datos desde el nodo *end-device* basado en el CC2520 hacia el coordinador.

el consumo de corriente del nodo basado el el procesador *ZigBee* CC2480 al realizar repetidamente el escaneo de un canal sin recibir ninguna respuesta por parte del coordinador. En este caso la corriente media durante el proceso es de unos 9,3 mA, mientras que si el número de canales se aumenta hasta 16, el consumo medio se incrementa hasta 27.6 mA, debido a que el dispositivo permanece casi todo el tiempo escaneando algún canal.

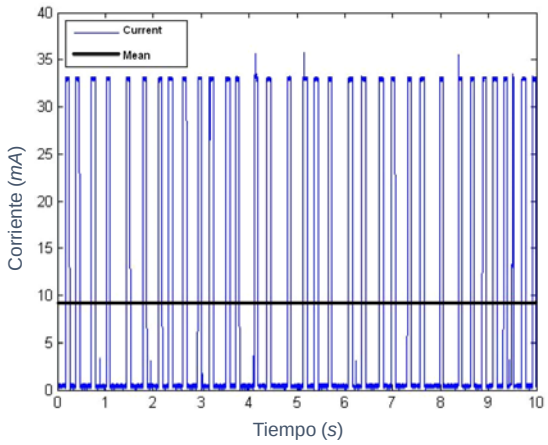


Figura 3.20: Corriente consumida por el dispositivo basado en el CC2480 durante la operación *orphan scan*.

3.4.2.5. Consumo durante el modo de bajo consumo

El consumo de los dispositivos en los periodos de inactividad es crucial en la duración de la vida de la carga de la batería, ya que en condiciones de operación ideales el dispositivo permanecerá en este modo la mayor parte del tiempo. Puesto que la corriente en este modo suele ser baja, la mejor forma de medirla es forzar a que el dispositivo entre permanentemente en este modo y utilizar una configuración del multímetro con periodos

largos de integración y un rango de medidas reducido. Las medidas recogidas para los diferentes dispositivos se recogen en el resumen presentado en las siguiente sección

3.4.2.6. Resumen de resultados

Los valores de consumo de corriente medido en las diferentes operaciones 802.15.4/*ZigBee* de los dispositivos basados en el procesador *ZigBee* CC2480 y en el transceptor CC2520 se muestran en las tablas 3.1 y 3.2 respectivamente.

Operación	**Estado**	**Corriente media**	**Duración**
Inactividad	Modo *Sleep*	0,75 μA	Variable
Inicio	Arranque del microcontrolador	2 mA	1,1 s
	Periodo de espera (Microcontrolador y ZNP activos)	15,5 mA	Variable
Asociación	escaneo en un canal	26,6 mA	2 s
	Escaneo en 16 canales	33,8 mA	up to 27,5 s
Envío de un paquete	Transmisión paquete de n bytes	30,5 mA	$1,06 + (8 \cdot n)/250$
	Escucha en el canal: Espera CSMA, CCA, Recepción de ACK	32,5 mA	2.9 ms
	Activación/desactivación del ZNP (transceiver radio apagado)	13 mA	13 ms
Pérdida de conexión	Escaneo en un canal	9,3 mA	Variable
	Escaneo en 16 canales	27,6 mA	Variable

Tabla 3.1: Consumo de las diferentes operaciones para el nodo basado en el CC2480.

Operación	**Estado**	**Corriente media**	**Duración**
Inactividad	Modo *Sleep*	0,30 μA (sólo transceptor)	Variable
Asociación al coordinador	Escaneo en un canal	17 mA	1,5 s
Envío de un paquete	Transmisión paquete de n bytes	26,5 mA	$1,06 + (8 \cdot n)/250$
	Escucha en el canal: Espera CSMA, CCA, Recepción de ACK	22,5 mA	2,3 ms
	Activación/desactivación del transceptor radio	1,8 mA	1,3 ms
Pérdida de conexión	Escaneo en un canal	6,5 mA	Variable (picos de 22,5 mA con 75 ms de duración)

Tabla 3.2: Consumo de las diferentes operaciones para el nodo basado en el CC2520 (sólo transceiver).

Estas medidas realizadas para las plataformas *ZigBee* de Texas Instruments han sido también repetidas en los mismos escenarios utilizando un sistema de desarrollo de Freescale formado por las placas 1322x-SRB (Sensor Reference Board) y 1322x-NCB (Network Coordinator Board), ambos basados en la plataforma MC1322x. Esta plataforma es un

SoC que incluye tanto la radio como la CPU que ejecuta el *firmware* que implementa la pila de protocolos y la aplicación, por lo que aunque en las mediciones se aisló el consumo del SoC del resto de componentes de la placa, no es posible aislar el consumo de la CPU y el de la radio. Los resultados se resumen en la tabla 3.3. Los resultados obtenidos con este sistema son similares a los anteriores, aunque este dispositivo presenta una diferencia en el comportamiento durante las esperas CSMA, en las que el interfaz radio permanece inactivo durante los *slots* de *backoff* y sólo se activa al final para realizar el CCA durante la duración mínima permitida por la norma. Esto permite ahorrar una cantidad de energía que puede llegar a ser importante si se producen muchas colisiones. Las últimas versiones del firmware ZStack disponible para el sistema basado en el CC2520 también implementan éste comportamiento.

Operación	**Estado**	**Corriente media**	**Duración**
Inactividad	Modo *Sleep*	0,3 μA	Variable
Asociación al coordinador	Escaneo en un canal	15.56 mA	2.85 s
Envío de un paquete	Transmisión paquete de n bytes	32 mA	$1,06 + (8 \cdot n)/250$
	Escucha en el canal: Espera CSMA, CCA, Recepción de ACK	25.5 mA	0.68 ms
	Activación/desactivación del transceptor radio	10 mA	1.2 ms
Pérdida de conexión	Escaneo en un canal	21.8 mA	Variable

Tabla 3.3: Consumo de las diferentes operaciones para el nodo basado en el MC1322x.

3.4.3. Estimación de la duración o vida útil de la carga de la batería

En esta sección se formula un modelo simplificado para estimar el tiempo de vida de un sensor que transmite periódicamente información hacia la red *ZigBee* y que fué publicado en [CCGCG10b].

3.4.3.1. Caso mejor: máxima duración de la batería

En la *testbed* experimental de la que se han obtenido los valores estimados para la corriente media extraída de la batería, sólo un dispositivo *end-device* interactúa con el coordinador, y tampoco se producen interferencias debido a dispositivos vecinos de otras tecnologías transmitiendo en la misma banda, por lo que no se producen colisiones ni se detecta ocupación de canal. En estas condiciones, el retardo introducido por el mecanismo CSMA/CA se ve reducido al mínimo y no se generan retransmisiones. Utilizando los valores medidos es posible realizar una estimación optimista de la duración de la carga de la batería que debe ser considerado como una cota superior.

Tras esperar el tiempo aleatorio impuesto por el algoritmo CSMA/CA, el tiempo necesario para transmitir una trama con un *payload* de n bytes se puede calcular como:

$$t_{tx}(n) = \frac{8 \cdot (O_{MAC/ZB} + n)}{r} \tag{3.8}$$

Donde r es la tasa de transmisión de datos de 802.15.4, 250 kb/s si la frecuencia de operación es un canal de la banda de 2,4GHz, y $O_{MAC/ZB}$ es la sobrecarga total de la trama 802.15.4/*ZigBee*, incluyendo el preámbulo, los delimitadores de trama, la cabeceras MAC y *ZigBee* y el campo de control de errores. En un paquete *ZigBee* correspondiente a datos de nivel de aplicación de un nodo que ya está asociado a la red y que por tanto utiliza la direcciones cortas en las cabeceras, esta sobrecarga es de 33 bytes, de los que 6 corresponden a la capa física, 11 al nivel MAC y 16 a los niveles de red y APS de *ZigBee*. Utilizando este tiempo, y en base a las medidas presentadas en la subsección 3.4.2.3, se puede estimar la corriente media que debe ser suministrada al dispositivo para realizar la transmisión de un paquete de n bytes como:

$$I_{active}(n) = \frac{t_{onoff}I_{onoff} + t_{listening}I_{listening} + t_{tx}(n)I_{tx}}{t_{act}(n)} \tag{3.9}$$

Donde t_{onoff} e I_{onoff} son el tiempo total y la corriente media necesarios para despertar y apagar el transceptor, así como enviar los datos hacia el microcontrolador, $t_{listening}$ e $I_{listening}$ corresponden por su parte con el tiempo medio y la corriente media necesarios para acceder al canal radio (utilizando CSMA/CA) y recibir el correspondiente ACK desde el dispositivo coordinador 802.15.4 (2,3 ms y 22,5 mA respectivamente en el caso medio observado para el CC2520). De forma similar, I_{tx} representa la corriente media consumida durante la transmisión de la trama de n bytes. Finalmente, $t_{act}(n)$ indica el tiempo total del periodo de actividad completo:

$$t_{act}(n) = t_{onoff} + t_{listening} + t_{tx}(n) \tag{3.10}$$

Si los datos son enviados de forma periódica con un periodo de actualización (tiempo entre dos transmisiones consecutivas) T, la corriente media que se extrae de la batería se podría calcular mediante la siguiente expresión:

$$I_{drain}(n) = \frac{t_{act}(n)}{T}I_{active}(n) + \left(1 - \frac{t_{act}(n)}{T}\right) \cdot I_{sleep} \tag{3.11}$$

Donde I_{sleep} es la corriente media mientras el dispositivo permanece inactivo y $\left(\frac{t_{act}(n)}{T}\right)$ representa el ciclo de trabajo del nodo sensor.

Despreciando el consumo realizado durante la fase de inicialización y asumiendo que todo el tráfico es ascendente y que no se realizan sondeos como parte de la operación del dispositivo *end-device*, la vida en años de una batería de capacidad C (medida en mAh), se podría derivar de I_{drain}:

$$L = \frac{C/(365 \cdot 24)}{I_{drain}(n)} \tag{3.12}$$

Las figuras 3.21 y 3.22 muestran la estimación de I_{drain} y L para diferentes valores de periodicidad de envío de los datos en el rango de 0,1 a 16 s. Las curvas consideran una capacidad de 1.200 mAh, que puede considerarse un valor típico para las baterías AAA que suelen emplearse para alimentar los nodos. En cada figura se incluyen los valores para dos casos extremos de tamaño de los datos enviados: 2 y 80 bytes. La mayoría de las

magnitudes que pueden ser medidas en una aplicación práctica para una red de sensores requerirán habitualmente un número de bytes comprendido entre estos dos extremos (siendo normalmente suficiente con unos pocos bytes). En tal caso, las figuras muestran que la máxima duración de la carga de la batería prevista para un dispositivo sensor en una red *802.15.4/ZigBee* podría llegar a ser de hasta varios años (en caso de un comportamiento ideal) en un escenario típico.

Como ya se ha mencionado previamente, debido a las condiciones ideales que se han supuesto, los tiempos de vida útil de la carga de la batería obtenidos en esta sección deben considerarse una cota superior. En las siguientes secciones se extiende el modelo de consumo (redefiniendo los valores de $t_{listening}$ y $t_{tx}(n)$) para añadir el efecto de las retransmisiones y de la reasociación de dispositivos que pueden tener lugar cuando se producen pérdidas de paquetes.

3.4.3.2. Caso peor: duración mínima de la carga de la batería (sin reasociación)

Como ya se ha explicado detalladamente en la la sección 3.2.4, los nodos que compiten por el acceso al canal siguiendo el algoritmo CSMA/CA deben aplicar inicialmente un tiempo de *backoff* esperando un número aleatorio de periodos de 0,32ms (20 símbolos) comprendido entre 0 y $(2^{BE} - 1)$, siendo BE un valor creciente denominado *backoff exponent* que alarga el tiempo de espera medio cuando el canal está saturado. Transcurrido este tiempo, el nodo comprueba la ocupación del canal (CCA), incrementando BE en una unidad y repitiendo la espera si está ocupado. Si el canal no se encuentra libre tras un número de reintentos, se considera que ha ocurrido un fallo de acceso al canal, que se indica a los niveles superiores, y se descarta el paquete a enviar. Si, por el contrario, en alguno de los intentos se estima que el canal está libre, el transceptor cambia del modo recepción al modo de transmisión y se realiza el envío del paquete. La transmisión se considerará correcta en caso de recibir correctamente el paquete de confirmación (ACK) que es enviado por el destino en cuanto se completa la recepción del paquete de datos (el algoritmo CSMA/CA no se aplica al paquete de confirmación). Si no se recibe el paquete de confirmación transcurrido el intervalo necesario para ello, se interpreta como una señal de que se ha producido una colisión debido a la actividad de nodos vecinos o a la interferencia de otras comunicaciones operando en la misma banda, y se reintenta el envío de paquete reiniciando de nuevo la operación del CSMA/CA. El número de reintentos está también limitados en este caso por el estándar y cuando se alcanza el máximo se cancela la transmisión del paquete y se informa del error a los niveles superiores.

En consecuencia, el caso peor de tiempo (y corriente consumida) necesario para completar el envío es aquel en el que el paquete es finalmente confirmado únicamente en el último intento de transmisión y tras repetir (en cada intento) la operación de CCA el máximo número de veces, empleando en cada una el mayor tiempo posible de *backoff* al realizar las esperas aleatorias.

En este caso peor, los tiempos de escucha del transceptor radio (es decir, el tiempo máximo que permanece en recepción) se pueden calcular analíticamente como:

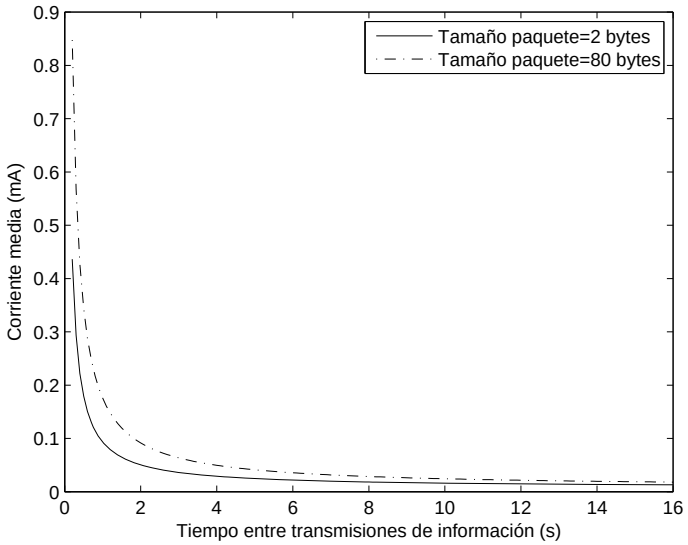


Figura 3.21: Consumo medio de corriente estimado (en condiciones ideales) para el dispositivo sensor en función de la tasa de envío para diferente tamaños de datos.

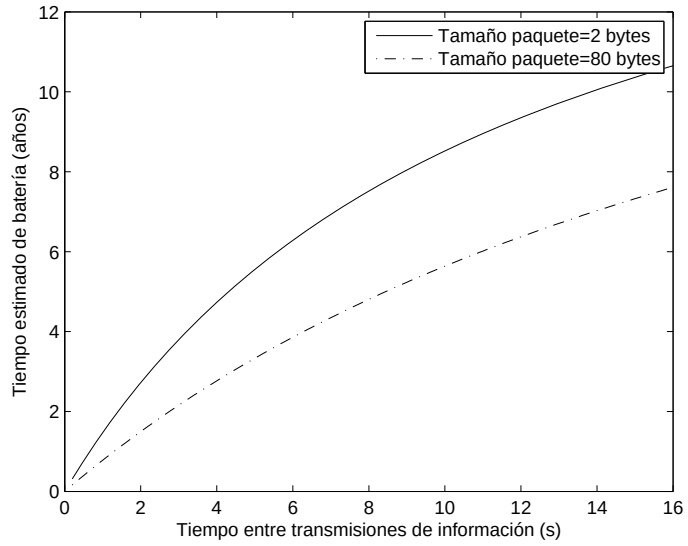


Figura 3.22: Duración esperada (en condiciones ideales) de la carga de batería de un dispositivo sensor en función de la tasa de envío para diferente tamaños de datos.

$$t_{listening} = (aMaxF + 1)(t_{CSMA(max)} + t_{TA} + t_{ACK}) \tag{3.13}$$

Donde $aMaxF$ es el máximo número de veces que se puede reintentar la transmisión del paquete (en el estándar el valor por defecto de éste parámetro, $macMaxFrameRetries$, es 3), t_{TA} es el tiempo reservado para que el transceptor conmute de recepción a transmisión o viceversa (12 símbolos o 0,192ms), mientras que t_{ACK} es el tiempo máximo durante el que se espera la llegada del paquete ACK de confirmación tras el envío de los datos (0,864ms o 54 símbolos). El término $t_{CSMA(max)}$ describe el máximo retardo que puede introducir el algoritmo de *backoff* del CSMA/CA y las operaciones de CCA en cada intento de transmisión. En el análisis del consumo realizado en las secciones anteriores para dispositivos comerciales se ha detectado que el firmware presente en muchas implementaciones mantiene la radio en modo recepción durante los periodos de espera del CSMA/CA, y por tanto este término se ha agregado en el cálculo del tiempo de escucha. De esta forma, el término $t_{CSMA(max)}$ puede ser calculado en estas implementaciones como:

$$t_{CSMA(max)} = \sum_{i=0}^{mMaxb} \left((2^{min(macMinBE+i, macMaxBE)} - 1) \cdot t_{backoff} + t_{CCA} \right) \tag{3.14}$$

Donde $mMaxb$ es el máximo número de veces que se repite el algoritmo CSMA tras el primer fallo de CCS. De acuerdo con el estándar el valor por defecto para este parámetro (denominado $macMaxCSMAbackoffs$) es 4. Los términos $macMinBE$ (3 por defecto) y $macMaxBE$ (5 por defecto) son los valores inicial y máximo del parámetro *backoff exponent* (BE) respectivamenteely, mientras que t_{CCA} es el tiempo necesario para realizar la operación de CCA (8 símbolos o 0,128 ms), y $t_{backoff}$ es la duración base de un periodo de *backoff* (20 símbolos o 0,32 ms). Asumiendo la configuración por defecto para todos los parámetros, se obtiene que $t_{CSMA(max)}$ y $t_{listening}$ son 37,44 ms (2.340 símbolos) y 153,98 ms (9.624 símbolos), respectivamente. En aquellas implementaciones que optimicen el comportamiento apagando la radio durante los periodos de *backoff*, sólo habría que tener en cuenta el tiempo debido a los CCA en el cálculo de $t_{CSMA)}$, que en dicho caso sería sensiblemente inferior ($t_{CSMA} = (mMaxb + 1) \cdot t_{CCA} = 0,64ms$).

Realizando un razonamiento similar, el cálculo del tiempo máximo que el transceptor permanece en el modo de transmisión requiere tomar en consideración que el paquete puede ser retransmitido hasta $aMaxF$ veces:

$$t_{tx}(n) = (aMaxF + 1) \cdot \left(\frac{8 \cdot (O_{MAC/ZB} + n)}{r} \right) \tag{3.15}$$

3.4.3.3. Caso medio: promedio de duración de la carga de la batería (sin reasociación)

El consumo medio de batería esperado depende evidentemente de la frecuencia con la que se produzcan las colisiones y los fallos de acceso al canal. Asumiendo que ambos mecanismos puedan modelarse como procesos estocástico independientes y autoincorrela-

dos, las ecuaciones definidas previamente para el caso peor pueden ser modificadas para estimar los tiempos de escucha y transmisión para el caso medio. En particular, si p_o denota la probabilidad de que el canal esté ocupado al realizar la operación de CCA y p_c es la probabilidad de que se detecte una colisión (es decir, de que el paquete ACK de confirmación no sea recibido) el tiempo medio de escucha en el que el transceptor radio permanece en modo recepción para realizar el envío de un paquete puede calcularse mediante la ecuación 3.16. Entre los trabajos relacionados citados anteriormente en la sección 3.3, [BCD+08] caracteriza empíricamente la probabilidad de error en función de la potencia de señal recibida y [GRX+11] ofrece una expresión analítica para calcular p_o y p_c en función del número de nodos contendiendo por el canal en una red *802.15.4/ZigBee* (sin otras fuentes de interferencia).

$$t_{listening} = \sum_{i=0}^{aMaxF} (1 - p_{CSMAfail})^i \cdot p_c^i \cdot \{p_{CSMAfail} \cdot t_{CSMAfail} + (1 - p_{CSMAfail}) \cdot (t_{CSMAnofail} + t_{TA} + t_{ACK})\} \quad (3.16)$$

Donde:

- $p_{CSMAfail}$ representa la probabilidad de sufrir un fallo de acceso al canal después de agotar los $mMaxb + 1$ intentos disponibles.

$$p_{CSMAfail} = p_o^{mMaxb+1} \quad (3.17)$$

- $t_{CSMAfail}$ describe la duración media de un intento de transmisión que acaba en un fallo de acceso al canal (tras $mMaxb + 1$ esperas CSMA y $mMaxb + 1$ operaciones CCA fallidas):

$$t_{CSMAfail} = \sum_{i=0}^{mMaxb} \left(\frac{1}{2} (2^{\text{mín}(macMinBE+i, macMaxBE)} - 1) \cdot t_{backoff} + t_{CCA} \right) \quad (3.18)$$

  Con los valores por defecto definidos por las especificaciones para *mMaxb, macMinBE* y *macMaxBE*, el resultado de las expresiones anteriores arroja un resultado de 19,04ms (1.190 símbolos) para $t_{CSMAfail}$.

- El término $r_{CSMAnofail}$ representa la estimación del retardo medio introducido por el algoritmo de *backoff* y las operaciones CCA en un intento de transmisión que no finaliza en un fallo de acceso al canal (es decir, que finaliza con un CCA en el que se detecta el canal libre). Este tiempo puede ser calculado [GRX+11] mediante la expresión:

$$t_{CSMAnofail} = \left(\frac{1-p_o}{1-p_o^{mMaxb+1}}\right) \cdot \sum_{i=0}^{mMaxb} p_o^i \left\{\sum_{j=0}^{i}\left(\frac{1}{2}(2^{\text{mín}(macMinBE+j,macMaxBE)}-1)\cdot t_{backoff}+t_{CCA}\right)\right\} \quad (3.19)$$

Por otra parte, el tiempo medio que el transceptor radio permanece en transmisión (para un paquete con un *payload* de datos de n bytes es:

$$t_{tx}(n) = \left(\frac{8\cdot(O_{MAC/ZB}+n)}{r}\right) \cdot \sum_{i=0}^{aMaxF}(1-p_{CSMAfail})^{i+1}\cdot p_c^i \quad (3.20)$$

Donde el término $\sum_{i=0}^{aMaxF}(1-p_{CSMAfail})^{i+1}\cdot p_c^i$ representa el número medio de veces que el paquete es retransmitido.

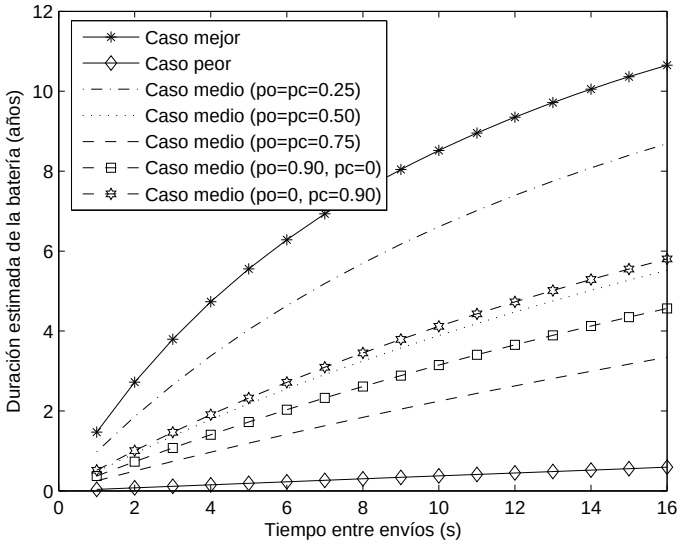
En la figura 3.23 se compara la duración prevista de la batería en los diferentes casos analizados (asumiendo de nuevo una capacidad de 1.200 mAh) para dos tamaños diferentes del *payload* de datos (2 y 80 bytes). Para el caso promedio, se muestran los resultados para diferentes probabilidades de colisión y fallo de CCA, p_c y p_o. Las gráficas ilustran importantes divergencias entre las diferentes situaciones contempladas, ya que el escenario en caso peor es excesivamente pesimista incluso frente a escenarios con elevadas probabilidades de fallo de CCA y colisión.

3.4.3.4. Efecto de la reasociación de los nodos en el consumo de corriente

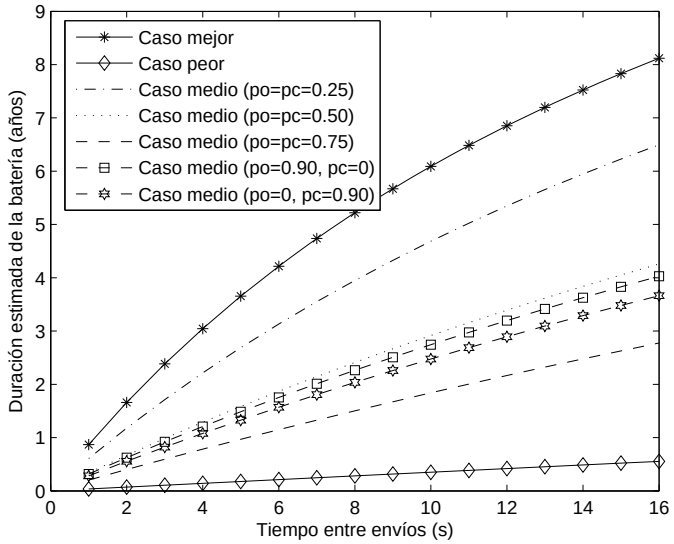
En las subsecciones anteriores se ha calculado el consumo de batería considerando que los nodos se asocian a la red *802.15.4/ZigBee* una única vez durante todo el tiempo de operación, justo tras el arranque inicial, y asumiendo por tanto que la carga extraída de la batería durante el escaneo activo, la asociación y la vinculación es despreciable para la estimación del tiempo de vida útil de la carga de la batería. Sin embargo, en la práctica, tras una pérdida de paquete repetida el nodo estará programado para realizar una nueva búsqueda e intentar reasociarse con un coordinador (si no está implementado el *orphan scan* o no se recibe el comando de realineamiento desde el coordinador tras el *orphan scan*). Como se mostró en la sección 3.2.4.5, este proceso de reasociación, que incluye el escaneo activo de canal y el intercambio de paquetes con el coordinador para proceder a la asociación a la red y a restablecer la vinculación, puede durar varios segundos con un consumo de corriente medio importante. En consecuencia, las pérdidas de paquete pueden introducir un consumo adicional que debe ser incorporado al modelo propuesto. En particular, el intervalo de actividad requerido para transmitir un paquete debe ser incrementado como sigue:

$$t_{act}(n) = t_{onoff} + t_{listening} + t_{tx}(n) + p_f \cdot t_{reassoc} \quad (3.21)$$

Donde $t_{reassoc}$ es el tiempo necesario para llevar a cabo el proceso de reasociación completo, mientras que p_f describe la probabilidad de que se produzca una pérdida del



a) Tamaño de datos (*payload* aplicación) 2 bytes



b) Tamaño de datos (*payload* aplicación) 80 bytes

Figura 3.23: Comparativa de la duración esperada de la carga de la batería para los distintos casos analizados y diferentes probabilidades de fallo de CCA p_o y colisión p_c (para una capacidad de 1200 mAh).

paquete, que puede calcularse [GRX+11] en función de las probabilidades de colisión p_c y de fallo de acceso al canal $p_{CSMAfail}$ (definido en la ecuación (3.17)), como:

$$p_f = (1 - p_{CSMAfail})^{aMaxF+1} \cdot p_c^{aMaxF+1} + \sum_{i=0}^{aMaxF} \left(p_{CSMAfail} \cdot (1 - p_{CSMAfail})^i \cdot p_c^i\right) \tag{3.22}$$

Donde $aMaxF$ es el parámetro de configuración ya mencionado que regula el número máximo de intentos de retransmisión.

De forma similar, la corriente media extraída de la batería cuando el transceptor no está inactivo, $I_{active}(n)$, debe ser redefinida como:

$$I_{active}(n) = \frac{t_{onoff} I_{onoff} + t_{listening} I_{listening} + t_{tx}(n) I_{tx} + p_f \cdot t_{reassoc} I_{reassoc.}}{t_{act}(n)} \tag{3.23}$$

Donde $I_{reassoc}$ es un nuevo término que define la corriente media requerida durante la fase de reasociación. Los valores concretos de $I_{reassoc}$ y $t_{reassoc}$ dependen obviamente del número de canales escaneados y de la probabilidad de sufrir pérdidas también durante el proceso de reasociación. Si consideramos el caso más favorable (un único canal y asumiendo que la reasociación es siempre exitosa), los valores caracterizados en la sección 3.4.2.4 para el CC2520 son 22,5 mA y 75ms respectivamente.

La figura 3.25 muestra los resultados considerando las ecuaciones del caso promedio para $t_{listening}$ y $t_{tx}(n)$. La figura 3.25.a compara el consumo de batería considerando y sin considerar el efecto de la reasociación tras la pérdida de paquetes, para dos probabilidades diferentes de pérdida de paquetes. Esta probabilidad se calcula en función de los valores de p_o y p_c considerados para estimar el consumo de corriente en el caso promedio. Por otra parte, la figura 3.25.b muestra la evolución del peso relativo del consumo de corriente debido a la reasociación en la vida media de la batería en función de la probabilidad de pérdida de paquete, para una tasa de 0,1 paquetes por segundo. Para calcular esta evolución, p_o y p_c han sido configurados al mismo valor y modificados desde 0,01 hasta 0,5 en incrementos de 0,01.

3.4.3.5. Impacto del sondeo de los dispositivos *end-device*

En los modelos de consumo planteados hasta ahora se ha despreciado el hecho de que los dispositivos *end-device* están obligados a sondear periódicamente a su dispositivo coordinador, de forma que éste pueda llevar a cabo las transmisiones indirectas necesarias para transmitir al *end-device* información procedente de la red. Este comportamiento es necesario incluso aunque el dispositivo se trate de un nodo sensor que sólo envía datos hacia la red, ya que es necesario para poder recibir la información de gestión de la misma. La periodicidad mínima con la que los dispositivos deben realizar un sondeo a su coordinador es un parámetro de configuración.

Empíricamente se ha comprobado que cuando el tiempo entre envíos de paquetes de datos es inferior al periodo de sondeo configurado, el dispositivo *end-device* no envía paque-

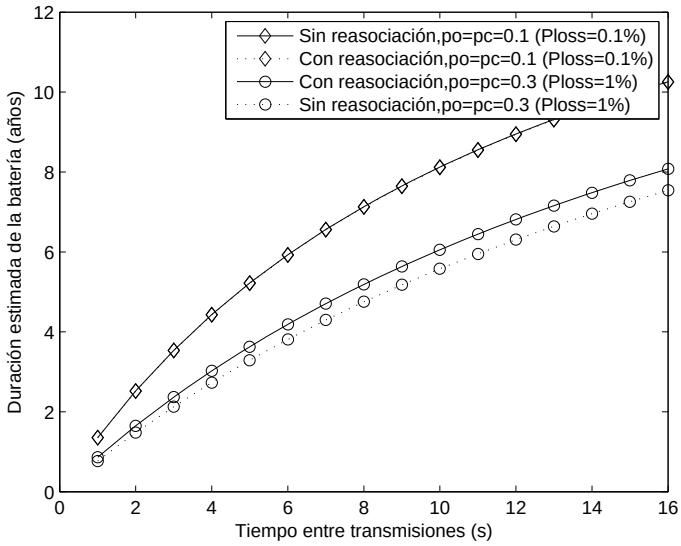


Figura 3.24: Impacto de la reasociación en la duración mínima de batería prevista (caso peor).

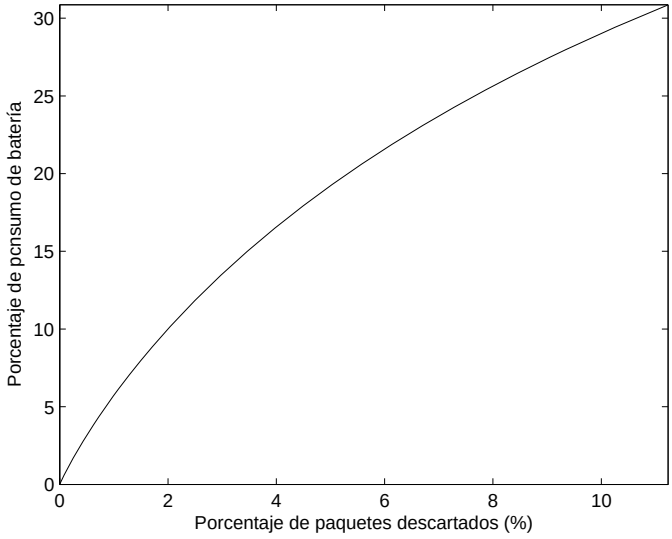


Figura 3.25: Impacto de la reasociación en el consumo de batería previsto (caso promedio para 2 bytes de *payload*).

tes para sondear a su coordinador (tramas MAC de control con el comando *Data Request*), ya que el coordinador puede indicar que tiene datos disponibles en el ACK de respuesta a los paquetes de datos. Las tramas *Data Request* únicamente se envían cuando el periodo de sondeo configurado es menor que el tiempo entre envíos de datos.

Así pues, en éste último caso sería necesario modificar los modelos propuestos para tener en cuenta el envío de los paquetes de sondeo. Desde el punto de vista del comportamiento energético éstos paquetes son similares a los de datos, pues se envían también utilizando el mecanismo CSMA/CA, pero su tamaño es menor (18 bytes). Por ello, se podrían utilizar aproximadamente las mismas expresiones, pero calculando una nueva tasa (o periodo) efectivo de envío de paquetes y un nuevo tamaño efectivo del paquete que refleje el tamaño medio de los paquetes (de un tipo u otro) enviados. Puesto que los paquetes de datos se envían siempre y los de sondeo sólo se envían cuando no se ha realizado ningún contacto con el coordinador transcurrido un tiempo T_{poll}, el número de paquetes de sondeo que se envían por cada paquete de datos es

$$N_{poll} = \left\lceil \frac{T}{T_{poll}} \right\rceil - 1 \tag{3.24}$$

El tiempo efectivo (para el cálculo del consumo) de envío de paquete será:

$$T_{eff} \approx \frac{T}{N_{poll} + 1} \tag{3.25}$$

Y el tamaño efectivo:

$$S_{eff}(n) \approx \frac{S_{poll} \cdot N_{poll} + O_{MAC/ZB} + n}{N_{poll} + 1} \tag{3.26}$$

Las ecuaciones desarrolladas en los apartados anteriores seguirían siendo válidas para este caso simplemente sustituyendo $(O_{MAC/ZB} + n)$ por S_{eff} y T por T_{eff}.

En los escenarios de prueba que se han montado con dispositivos comerciales, se ha comprobado que el tiempo de sondeo puede aumentarse hasta 5 segundos sin que se produzcan problemas con la configuración *ZigBee* por defecto. Para valores mayores, se hace necesario ajustar la configuración para evitar problemas de *timeout*.

3.4.4. Pruebas de validación del modelo

Para analizar la validez el modelo analítico de consumo que se propuso en [CCGCG10b] y recogido en la sección anterior, se han ejecutado experimentos reales sobre el sistema de prueba descrito en la sección 3.4.1. En estos experimentos se ha configurado el multímetro para realizar estimaciones de la corriente media utilizando tiempos de integración largos y promediando las medidas obtenidas a lo largo de un periodo de 30 minutos para cada escenario.

La figura 3.26 muestra el consumo estimado (utilizando el modelo) y el consumo medido (promediando con el multímetro) para el transceiver CC2520 en función del tiempo entre paquetes en un escenario con $T_{poll} = 5s$, en ausencia tanto de colisiones como de fallos de CCA, y en el que los paquetes de datos contienen 2 bytes de *payload* de nivel de

aplicación. El consumo medido y calculado corresponde únicamente a la operación del transceptor C2520, y no se ha considerado en ellos, ni se incluye en la medida, la operación del microcontrolador (que realiza otras operaciones además de la comunicación *ZigBee*). La figura 3.27 muestra los mismos valores para un escenario similar, en este caso en función del tamaño del *payload* de datos de nivel de aplicación enviados, para un tiempo entre paquetes de 0,2s (5 paquetes por segundo). En ambos casos se aprecia que el modelo propuesto se ajusta bastante bien al consumo de corriente medido en ausencia de pérdidas.

La figura 3.28 muestra un escenario en el que se han emulado pérdidas y fallos de CCA en el sistema de pruebas real, así como los valores estimados en función de los modelos propuestos, tanto el que incluye el efecto de la reasociación que puede producirse tras el descarte de un paquete por colisiones reiteradas, como el que no incluye este efecto. Para comparar los modelos con el comportamiento del sistema real se han realizado medidas utilizando dos configuraciones diferentes de la pila de protocolos ZStack. En una de ellas (círculos rojos), se han configurado los parámetros de la red *ZigBee* de forma que el nodo *end-device* se considera huérfano cada vez que se descarta un paquete debido a la colisión o fallo de CCA repetido (el paquete no se descarta en cualquier caso hasta fallar 4 veces su transmisión). Esta es la configuración más cercana a la ecuación propuesta en la subsección 3.4.3.4. En la otra configuración medida (cuadrados azules) se han ajustado los parámetros de la pila de protocolo de forma que el nodo *end-device* no se considera huérfano hasta que se hayan descartado 12 paquetes (cada uno de los cuales debido a 4 fallos de transmisión a nivel MAC). En este escenario en realidad la probabilidad de que el nodo se intente reasociar es muy baja, y sería un caso más cercano al modelo que no tenía en cuenta el efecto de la reasociación. En ambos escenarios se han desactivado las retransmisiones tanto en el nivel de red como en el de aplicación (sólo el MAC retransmite en caso de colisión).

Los resultados muestran que, para los valores de pérdidas medidos, el modelo que no tiene en cuenta el efecto de la reasociación a la red se ajusta bastante bien al consumo de colisiones y fallos de CCA en el escenario en que dichas reasociaciones no se producen, mientras que el modelo que tiene en cuenta el efecto de las mismas parece sobreestimarlo en comparación a las medidas obtenidas en los escenarios en los que sí se producen dichas reasociaciones. Esto es así porque antes de realizar un intento de reasociación el dispositivo *end-device* intenta resincronizarse con su coordinador mediante el envío de un paquete de *orphaning* que puede tener éxito, y sólo procede a reescanear los canales en busca del coordinador cuando el procedimiento anterior falla.

Por último, la figura 3.29 muestra el consumo medio de corriente cuando el periodo de sondeo (T_{Poll}) configurado para las transmisiones indirectas es menor que el tiempo entre paquetes. La prueba ha sido realizada utilizando un *payload* de 80 bytes y un periodo de envío de 2 segundos para los paquetes de datos, en ausencia de colisiones y fallos de CCA. La línea azul corresponde a los valores estimados mediante las fórmulas propuestas en la sección 3.4.3.5, y los puntos rojos a los valores medidos en el sistema de prueba.

3.4.5. Pruebas empíricas adicionales

En el estudio analítico realizado en la sección anterior se ha analizado principalmente el consumo debido a la actividad del MAC. Para validar el modelo se ha hecho uso de la

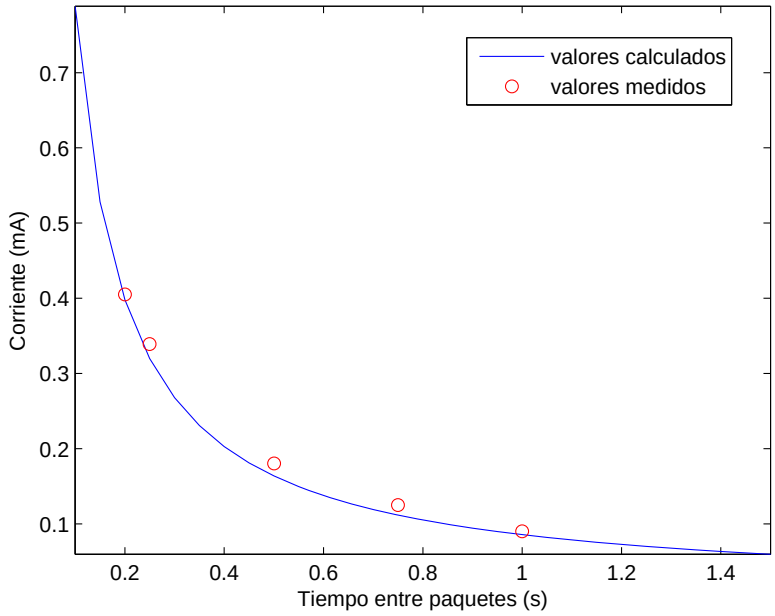


Figura 3.26: Consumo de corriente estimado y medido para diferentes tiempos entre paquetes (*payload* de 2 bytes datos, $T_{poll} = 5s$).

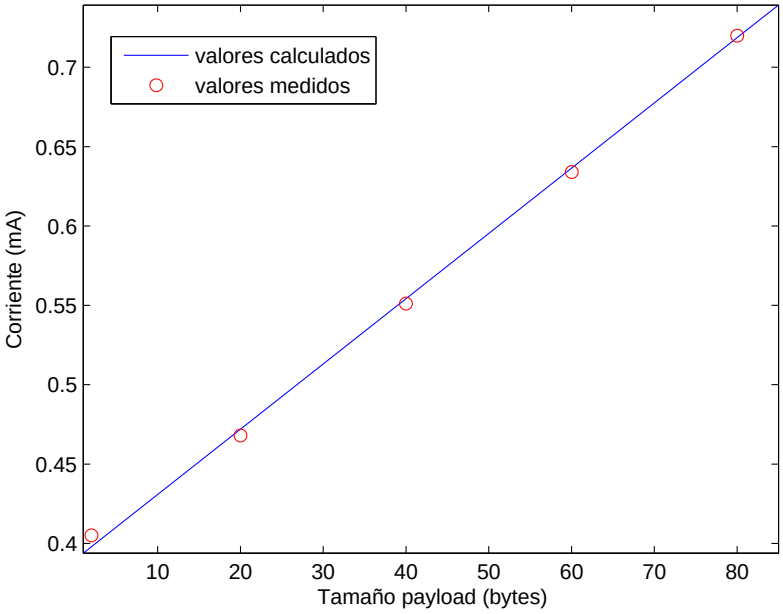


Figura 3.27: Consumo de corriente estimado y medido para diferentes tamaño de *payload* de datos ($T_{poll} = 5s$, tiempo entre paquetes $T = 0,2s$).

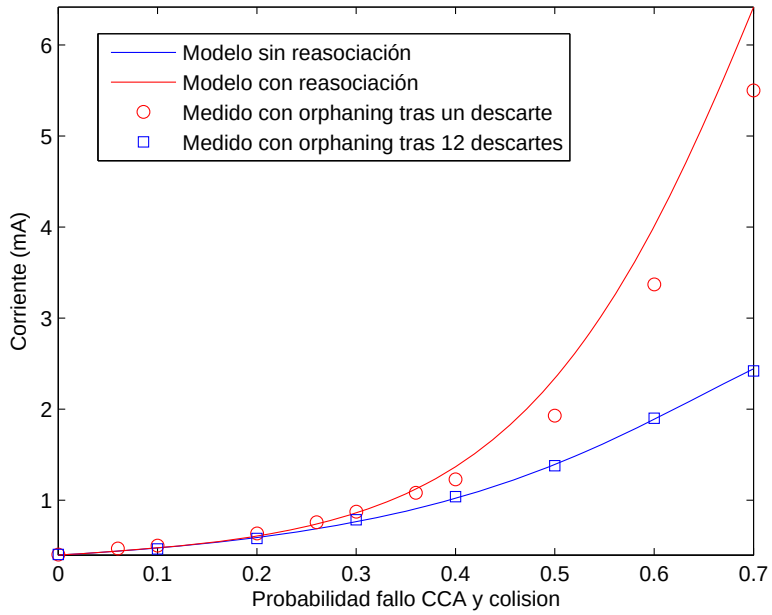


Figura 3.28: Consumo de corriente estimado y medido para diferentes probabilidades de colisión y fallo de CCA (*payload* de 2 bytes datos, $T_{poll} = 5s$, tiempo entre paquetes 0,2s).

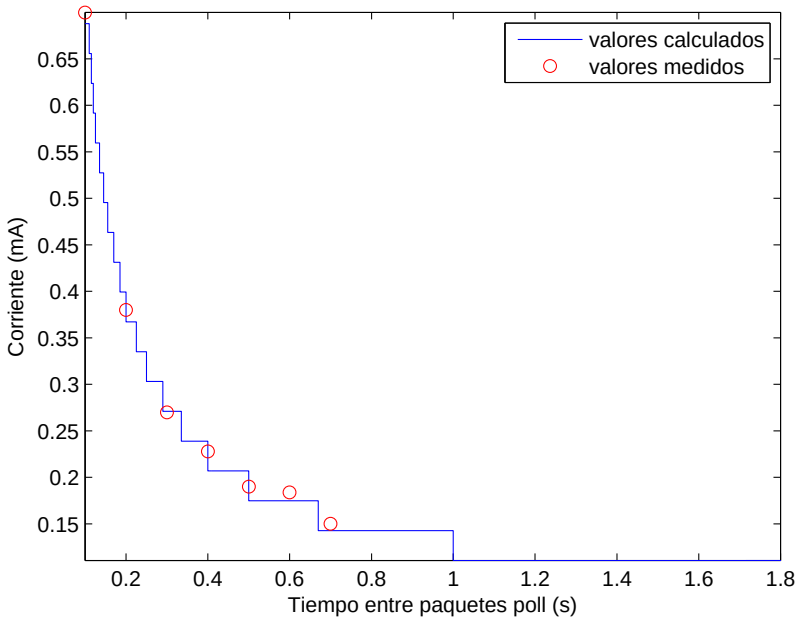


Figura 3.29: Consumo de corriente estimado y medido para diferentes periodos de sondeo T_{Poll} (sin colisiones ni fallos de CCA, paquetes de datos con 80 bytes de *payload* enviados cada 2s).

caracterización del transceptor y se ha medido el consumo medio del mismo en diferentes escenarios reales, y sin considerar el efecto del consumo del microcontrolador, que no sólo se activa debido a la operación de la capa MAC, sino que también debe hacerlo para el resto de capas. Para validar el modelo, además, se ha modificado la configuración de los parámetros de algunas de éstas otras capas con respecto a los valores por defecto establecidos por la implementación comercial de la pila de protocolos (muchos de estos valores no son definidos por la norma, pero sí por implementaciones comerciales).

En esta sección se incluyen algunas medidas adicionales, publicadas en [CCA12], que incluyen el consumo del microcontrolador y la configuración por defecto de parámetros de la pila de protocolos *ZStack*, que determinan que un paquete descartado por el MAC debido a las colisiones puede ser retransmitido una vez más por el nivel de red y que el dispositivo no se considera huérfano hasta haber perdido 4 paquetes de datos o 2 de *poll* por descarte a nivel MAC.

En este estudio además, se utilizó una versión más reciente de la pila de protocolos que sólo activa la recepción del transceptor para realizar el CCA durante el último *slot* de *backoff*, permaneciendo con la radio desactivada (transceptor inactivo, aunque no dormido) durante el resto del tiempo de *backoff* cuando éste es mayor a 1 *slot*.

La figura 3.30 muestra el consumo de corriente medido para diferentes probabilidades de fallo de CCA y colisión (y de ambos a la vez con igual probabilidad). En este escenario, se transmite cada 0,2s un paquete con 9 bytes de datos de nivel de aplicación, y el valor del tiempo de sondeo T_{poll} se configuró a 5s. Las medidas fueron obtenidas en las mismas condiciones mencionadas anteriormente, utilizando el multímetro con tiempos de integración largos y promediando las medidas durante 30 minutos (correspondientes a la transmisión de 9000 paquetes de datos) para obtener cada punto.

La figura muestra que las colisiones son mucho más perjudiciales para el consumo del dispositivo que el fallo de CCA (especialmente en este escenario de pruebas en el que la versión del *firmware* utilizado incorpora la mejora que permite reducir el consumo durante el *backoff*), ya que el descarte de paquetes por fallo repetido de CCA no produce que el dispositivo *end-device* se considere huérfano. En cambio, la combinación de fallos de CCA con las colisiones si aumenta considerablemente el consumo. Cuando la probabilidad de fallo de CCA y colisión es alta (algo por encima de 0,8), el consumo del dispositivo se dispara hasta 25,6mA debido a que los dispositivos se desconectan de la red y emplean casi todo el tiempo intentando volver a reasociarse.

Estos resultados alertan de que la presencia de determinadas fuentes de interferencia que puedan producir periodos sostenidos de alta probabilidad de detectar el canal ocupado o de perder el paquete durante su transmisión podrían afectar bastante a la duración de la batería de un nodo sensor.

3.5. Evaluación empírica del retardo

En el estudio incluido en las secciones anteriores se ha mostrado como el consumo de la red se ve claramente afectado por el tiempo de sondeo T_{poll} que determina el ciclo de actividad de los nodos *end-device* cuando no tienen información que enviar. Como es lógico,

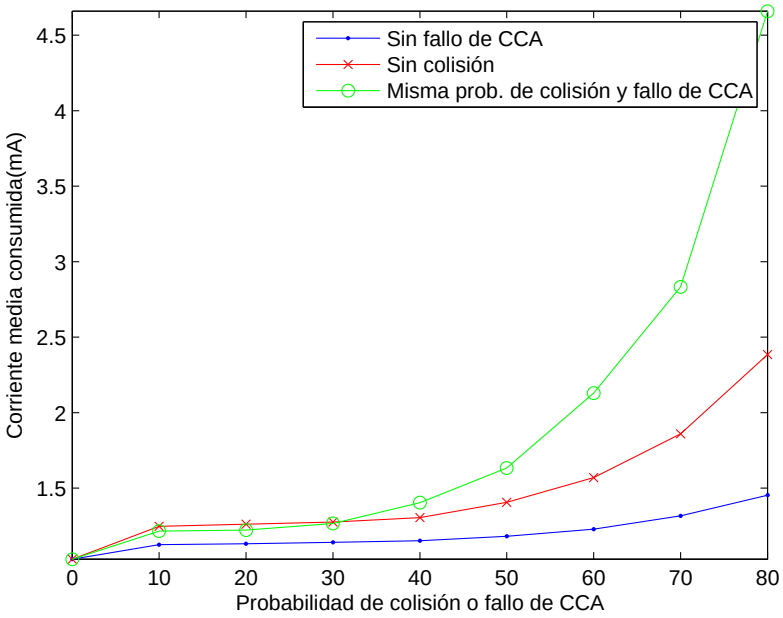


Figura 3.30: Consumo de corriente estimado y medido en función de la probabilidad de colisión y fallo de CCA.

la modificación de los parámetros de funcionamiento de la red puede afectar al rendimiento de la misma. Por ello, para completar la evaluación empírica de la tecnología *ZigBee* se ha desarrollado también una metodología para permitir la realización de medidas de retardo sobre sistemas de pruebas reales que se presenta en esta sección.

En el caso de *ZigBee*, el principal impedimento para realizar este tipo de medidas en plataformas reales es que habitualmente los dispositivos sensores disponibles (a diferencia de los dispositivos *Bluetooth* estudiados en el capítulo anterior) están basados en microcontroladores y no cuentan con un interfaz de alta velocidad que los interconecte con un PC (normalmente la conexión está basada en una conexión serie UART). En *ZigBee* además el retardo ascendente y descentente entre un dispositivo *router* y un dispositivo *end-device* es de naturaleza fuertemente asimétrica, ya que la comunicación desde el *end-device* hacia el *router* se realiza mediante transmisión directa, mientras que en sentido contrario la transmisión es indirecta. Una de las posibles alternativas para poder realizar una medida adecuada del retardo en cada sentido de transmisión (OTT) es establecer algún mecanismo que permita sincronizar con suficiente precisión los relojes de los nodos origen y destino del tráfico intercambiado.

Para realizar las medidas presentadas en esta sección se ha utilizado de nuevo la plataforma basada en el transceptor CC2520 con el microcontrolador MSP430F5438, y para conseguir la necesaria sincronización entre los nodos se han aprovechado algunos recursos del microcontrolador que no son utilizados por la pila de protocolos Z-Stack. En cada dispositivo, un temporizador *hardware* es utilizado para generar una base de tiempo de 1 segundo a partir del oscilador de 32,768 kHz controlado por cristal, que es la fuente de reloj de mayor precisión disponible en el microcontrolador. La interrupción del temporizador

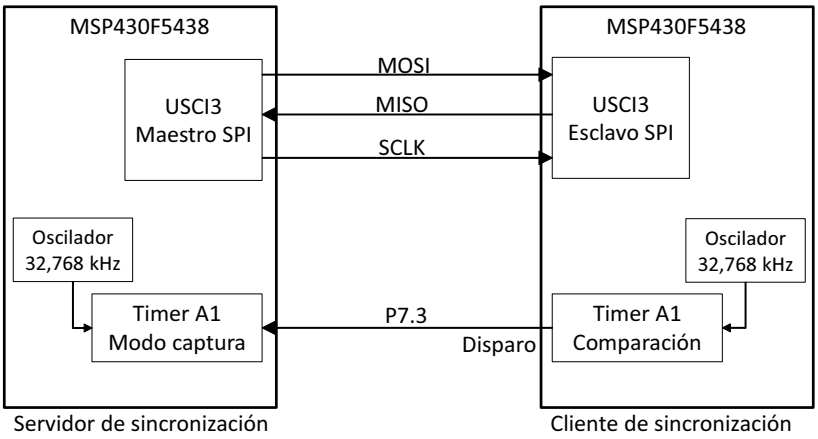


Figura 3.31: Configuración del sistema para la sincronización del reloj entre nodos

permite actualizar los segundos transcurridos, mientras que el valor de cuenta del propio temporizador (0 a 32767) representa la fracción por debajo del segundo.

Para que los dispositivos estén sincronizados es necesario que ambos compartan los mismos valores de cuenta del temporizador y de la cuenta de segundos. Para realizar esta sincronización se ha optado por utilizar el mecanismo de captura del temporizador y uno de los interfaces serie SPI disponibles para establecer una conexión cableada que permita intercambiar información de temporización, tal y como se esquematiza en la figura 3.31. La configuración no es simétrica, sino que uno de los dispositivos actúa como servidor de sincronización, mientras que el otro (cliente) le realiza peticiones de forma periódica para obtener información que le permita mantenerse sincronizado con él. El temporizador del dispositivo cliente se ha configurado de forma que genera cada segundo (y con un *offset* de 0,5 segundos con respecto a la interrupción que actualiza la cuenta de segundos) una flanco de disparo que provoca la captura *hardware* del valor de cuenta y una interrupción en el dispositivo servidor. Al producirse esta interrupción, el dispositivo servidor envía por el interfaz SPI el valor de cuenta capturado y el número de segundos hacia el dispositivo cliente, que al recibirlo actualiza adecuadamente la cuenta de segundos y el valor de cuenta del temporizador para compensar la diferencia existente entre ambos en el momento en el que se generó la señal de disparo.

Con este mecanismo se consigue mantener una sincronización con un margen de error inferior a 1/32,768 segundos ($30,5\mu s$) sin introducir apenas sobrecarga de procesado (ni por tanto alterar el funcionamiento de la pila de protocolos) en el microcontrolador. La sincronización además no se ve afectada por posibles latencias variables en el envío de la información por el interfaz SPI o en su procesado, ya que tanto la generación de la señal de disparo como la captura del valor de cuenta del temporizador se realizan por parte de recursos *hardware*. Lógicamente, el principal inconveniente de mecanismo propuesto es la necesidad de establecer una conexión cableada entre los dispositivos origen y destino.

Una vez garantizada la sincronización del reloj entre los dispositivos origen y destino, el retardo (OTT) de los paquetes puede ser medido mediante la inclusión de una marca de tiempo en el *payload* del paquete.

Las figuras 3.32 y 3.33 muestran los valores del retardo medidos en sentido ascendente (del *end-device* hacia el *router/coordinador*) y descendente (desde el *router/coordinador* hacia el *end-device*) para diferentes probabilidades de fallo de CCA y colisión (emuladas como ya se describió en la sección 3.4.1), en función de parámetros como el tamaño de datos del paquete y el tiempo de sondeo T_{poll}.

Los resultados representados en estas figuras corresponden a valores medios del retardo de paquetes enviados en un único sentido, utilizando un tiempo entre paquetes aleatorio para evitar posibles distorsiones debido a la sincronización con la operación del interfaz radio. En cada punto evaluado se ha obtenido la media de 5000 paquetes. Los paquetes han sido enviados sin confirmación extremo a extremo, por lo que no se retransmiten al ser descartados. Los intervalos de la distribución aleatoria que sigue el tiempo entre envíos de paquetes se han adaptado en cada configuración evaluada de forma que no se acumulen varios paquetes en las colas de transmisión (la acumulación de paquetes en las colas de transmisión reduciría el retardo medio en el enlace descendente al permitir la transmisión consecutiva de varios paquetes en cada sondeo, mientras que en el enlace ascendente se produciría un aumento del mismo debido al encolamiento de las transmisiones). El tráfico escogido se correspondería de forma realista con el de un dispositivo sensor que envía datos cada cierto tiempo o el de un dispositivo actuador que recibe algunas órdenes de forma ocasional.

Los resultados muestran la importante asimetría entre ambos sentidos de la comunicación, que viene dada por el empleo del mecanismo de transmisión indirecta en el sentido descendente para ahorrar energía en los nodos *end-device*. En este sentido de la comunicación el retardo viene principalmente determinado por el tiempo entre paquetes de sondeo y por la probabilidad de perder alguno de estos paquetes cuando las colisiones o los fallos de CCA ocurren con mayor probabilidad. En ausencia de pérdidas el retardo medio de los paquetes se aproxima a $T_{poll}/2$, lo que es lógico puesto que no se produce encolamiento de paquetes.

Los retardos en el sentido ascendente son muy inferiores, ya que se emplea el mecanismo de transmisión directa. Por ello, en este escenario sí resulta perceptible la variación del tiempo de transmisión en función del tamaño del paquete. La disminución de la pendiente ascendente de la curva en el último tramo (para probabilidades altas de fallo de CCA y de colisión) se debe a que un aumento de la probabilidad de fallo de CCA hace que el paquete se descarte sin llegar realmente a intentar transmitirlo.

3.6. Conclusiones

La tecnología 802.15.4/ZigBee es desde hace algunos años uno de los estándares más prometedores en el ámbito de las redes de sensores inalámbricas de bajo consumo. Tras un análisis detallado del funcionamiento de dicha tecnología, en este capítulo se ha realizado la caracterización del comportamiento energético de diversas implementaciones de dispositivos comerciales, considerando tanto las operaciones de transmisión de datos como algunas operaciones para el establecimiento y mantenimiento de la red.

Una vez realizada esta caracterización empírica del consumo, y considerando el com-

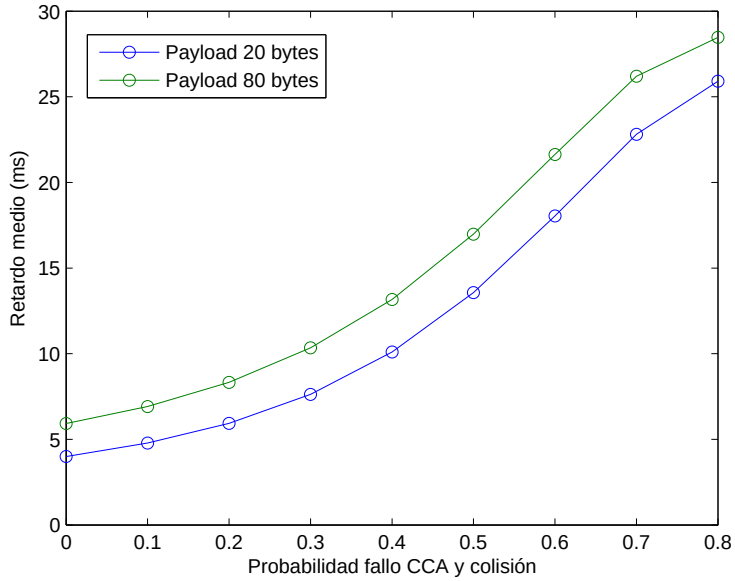


Figura 3.32: Retardo en el enlace ascendente medido en función de la probabilidad de colisión y fallo de CCA para diferentes tamaños de *payload*.

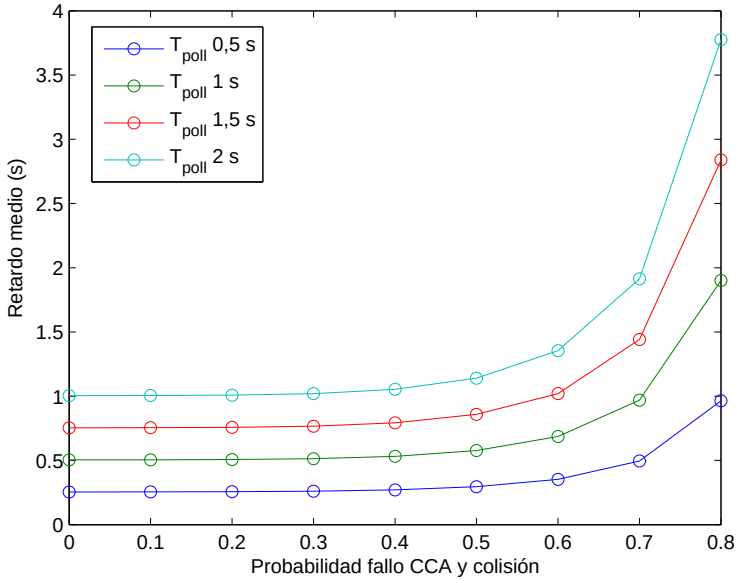


Figura 3.33: Retardo en el enlace descendente (transmisión indirecta) medido en función de la probabilidad de colisión y fallo de CCA y de T_{poll} (paquetes de 20 bytes de *payload*).

portamiento que tiene un dispositivo *end-device*, se han planteado una serie de expresiones matemáticas que permitirían estimar la duración de la batería de un dispositivo sensor en función del tamaño de los datos que envía y la periodicidad con la que lo hace. El modelo propuesto define ecuaciones para obtener la duración máxima, mínima y media esperada para la carga de la batería, teniendo en consideración las esperas introducidas por el algoritmo CSMA/CA aplicado por la capa MAC de 802.15.4.

El modelo propuesto ha sido validado mediante la realización de medidas en el sistema de pruebas real para diversas configuraciones, y se han presentado algunas medidas adicionales que extienden los casos previstos por el modelo.

Los resultados muestran que aunque en condiciones ideales el tiempo de operación a batería de un sensor *end-device* con un bajo ciclo de trabajo puede ser teóricamente largo, escenarios con un gran densidad de nodos y/o la presencia de fuentes de interferencias de otras tecnologías puede provocar una degradación apreciable en el mismo.

Capítulo 4

Evaluación empírica y modelado de WPAN basadas en *Bluetooth low energy*

4.1. Introducción

En este capítulo se recogen las aportaciones realizadas sobre la evaluación de redes de área personal con tecnología *Bluetooth low energy*. Siguiendo la misma estructura que en capítulos anteriores, en primer lugar se realiza una revisión detallada del funcionamiento de todos los niveles de la arquitectura de protocolos definida por el estándar derivada del estudio de sus especificaciones, en el que la sección 4.2.3.3 concentra los aspectos más relevantes que determinan el consumo de la tecnología.

Tras la descripción del funcionamiento, en la sección 4.3 se realiza una revisión de los diferentes campos de estudio en los que se centran los diferentes trabajos y propuestas realizados por la comunidad investigadora en relación con la tecnología, y por último en la sección 4.4 se incluyen las aportaciones realizadas dentro del marco de la presente tesis.

4.2. Fundamentos de *Bluetooth low energy*

La tecnología *Bluetooth Low Energy* es una modificación que afecta principalmente a los niveles más bajos de la pila de protocolos y que ha sido introducida en las especificaciones del *Bluetooth SIG* a partir de la versión 4.0 de las mismas. Anteriormente a su inclusión en el estándar, la tecnología *Bluetooth low energy* era conocida como *Wibree*, y actualmente se está empezando a denominar con el nombre comercial de *Bluetoot Smart*. Aunque también se han introducido nuevos perfiles de aplicación, las principales modificaciones han sido introducidas en la capa radio y de enlace con objeto de mejorar la eficiencia energética [DeC14]. La tecnología *low-energy* no es compatible hacia atrás con la tecnología *Bluetooth* tradicional, pero sí puede coexistir con ella y ambas pueden implementarse fácilmente en el mismo circuito integrado. Al igual que la tecnología *Bluetooth* original, la tecnología *low energy* está orientada a redes con una topología en estrella, pero está optimizada para que los dispositivos puedan operar con un ciclo de trabajo reducido. La arquitectura de

protocolos se muestra en la figura 4.1.

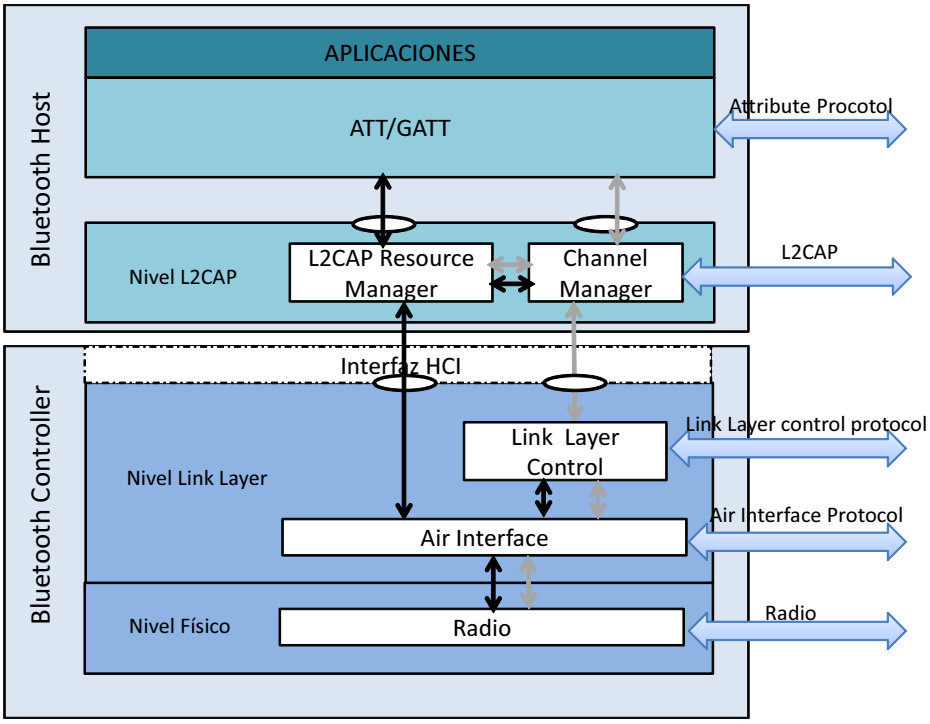


Figura 4.1: Arquitectura de protocolos *Bluetooth low energy*

Los nuevos perfiles desarrollados se basan en la gestión de atributos que es facilitada por un nuevo protocolo introducido en la pila, denominado ATT, y que permite que un dispositivo publique una serie de atributos que corresponden con su funcionamiento, y que pueden ser descubiertos, leídos o modificados remotamente por otros dispositivos. Los perfiles basados en esta implementación tienen la ventaja de minimizar el volumen de información que se tienen que intercambiar los dispositivos para soportar la aplicación, y que éstas pueden ser implementadas de forma más sencilla y con menor gasto de recursos.

4.2.1. Tipos de dispositivos y topologías

La norma *Bluetooth low energy* sigue definiendo la *piconet* como la topología básica de red, en la cual un dispositivo maestro se está comunicando con uno o varios dispositivos esclavos. Al igual que en la modalidad BR/EDR, los dispositivos esclavos únicamente se comunican con el maestro y no intercambian información directamente entre ellos, pero a diferencia de dicha modalidad, en *Bluetooth low energy* además los esclavos no comparten una sincronización común (y por tanto un enlace físico común) con el maestro, sino que cada uno mantiene un enlace físico separado con una sincronización distinta. El número de esclavos que un maestro puede soportar no está limitado a 7 por la especificación, sino que ésta permite que el maestro pueda llegar a establecer un número considerable de enlaces, lo que en la práctica conlleva que el tamaño de la *piconet* está principalmente limitado por los recursos de memoria y procesado disponibles en el dispositivo maestro. La formación de topologías complejas tipo *scatternet* queda específicamente descartada en la modalidad *low-energy*, y tampoco se permite el cambio de rol por parte de los dispositivos. Además, la

norma no indica claramente cómo debe el dispositivo maestro gestionar varios dispositivos esclavos, si bien los mecanismos especificados permiten que así sea.

Para permitir el descubrimiento de dispositivos y la formación de *piconets*, *Bluetooth LE* prevé una serie de procedimientos que son similares en funcionalidad a los previstos en la modalidad BR/EDR, pero que están diseñados para una mayor eficiencia energética. Además, el estándar define una serie de comportamientos para los dispositivos asociados a estos procedimientos. Para que un dispositivo sea visible por otros debe estar realizando el procedimiento de *advertising*, que consiste en la difusión de paquetes *broadcast* conteniendo determinada información sobre el dispositivo. Un dispositivo realizando este procedimiento recibe el nombre de *advertiser*. La información difundida por un dispositivo *advertiser* puede ser recibida por otros, lo que se utiliza tanto para el intercambio de información como para el descubrimiento de dispositivos y para el establecimiento de conexiones. El descubrimiento de dispositivos vecinos es necesario para la formación de redes espontáneas, y se realiza mediante el procedimiento de *scanning*, que básicamente consiste en permanecer a la escucha en los canales radio destinados al *advertising* y procesar los paquetes enviados por los dispositivos *advertisers*. Los dispositivos que realizan este procedimento de escucha reciben el nombre de *scanners*. El establecimiento de conexiones también requiere que los dispositivos que inician el procedimiento de conexión, denominados *initiators*, escuchen en los canales de *advertising* para sincronizarse con dispositivos *advertisers*. Estos procedimientos se describirán con mayor detalle en las siguientes secciones.

4.2.2. Capa Física

La capa física de *Bluetooth low energy* sigue utilizando una modulación GFSK, pero con un índice de modulación mayor que el empleado en el *Bluetooth* original (0,5 en lugar de 0,35), por lo que la modulación se aproxima bastante a un esquema GMSK, lo que permite reducir los requisitos de potencia radio [Fur10], mejorando por tanto el alcance. Con esta configuración, se requieren canales de 2 MHz de ancho de banda, el doble del empleado por la tecnología *Bluetooth* original para la misma tasa binaria nominal de 1 Mb/s. Por este motivo, el número de canales que se pueden crear en la banda ISM de 2.4 GHz se reduce, pasando de 79 a 40. Como más adelante se volverá a comentar, estos tres canales se dividen en 37 canales de datos (numerados del 0 al 36) y 3 canales de señalización (o *advertising*). Los 3 canales de señalización están separados en el espectro y se sitúan en aquellas frecuencias que se verían menos afectadas por los tres canales principales establecidos por 802.11, para reducir la interferencia con posibles redes coexistentes en la vecindad. Bluetooth low energy también utiliza un esquema de acceso múltiple basado en TDMA/TDD con salto de frecuencia, pero las restricciones de temporización, tanto para el TDMA como para el salto de frecuencia, son bastante más relajadas que en la tecnología *Bluetooth* original. Como más adelante se explicará con mayor detalle, los canales de señalización se utilizan tanto para el descubrimiento de disposisitivos en la vecindad como en el establecimiento de conexiones entre dispositivos mediante el envío de paquetes de *advertising*. Una vez establecida la comunicación y sus parámetros, ésta se lleva a cabo utilizando los canales de datos.

Al estar orientada a dispositivos con bajas necesidades de tasa de datos y restricciones de bajo consumo, las tramas radio de *Bluetooth LE* son bastante más cortas que las de la tecnología original, y constan de un preámbulo de 1 octeto, un código de acceso de 4 octetos, una PDU de longitud variable entre 2 y 39 octetos, y un CRC de 3 octetos. Esto arroja un tamaño mínimo de paquete de 80 bits y un máximo de 376 bits. La tasa binaria nominal de 1 Mbps permitiría transmitir la información 4 veces más rápido que en otras tecnologías competidoras basadas en 802.15.4, reduciendo el tiempo que el transmisor permanece encendido con respecto a éstas.

4.2.3. Capa de enlace (*Link-Layer*)

La capa *Link-Layer* realiza una función equivalente a las capas *Baseband* y *Link Manager* de la versión *Bluetooth* BR/EDR, definiendo los mecanismos para llevar a cabo los procedimientos de búsqueda de dispositivos, establecimiento de conexión, y comunicación entre maestro y esclavos de una piconet utilizando los canales radio definidos por la capa física.

Puesto que el funcionamiento de las comunicaciones en *Bluetooth low energy* requiere una sincronización entre los dispositivos que se comunican, para describir la temporización de los diferentes procedimientos el estándar establece determinados instantes de referencia llamados *eventos*, de los que define dos tipos: eventos de conexión, y eventos de *advertising*, de los que a su vez existen varios subtipos. Estos eventos definen cuándo se envían o se pueden enviar los diferentes tipos de paquetes radio, y la relación que el estándar establece entre ellos determina la temporización de las comunicaciones, como se irá detallando a lo largo de esta sección.

4.2.3.1. Formato y tipos de PDU

Como ya se ha mencionado en la sección 4.2.2, las tramas radio tienen una longitud variable y constan de un preámbulo, un código de acceso, una PDU de nivel de enlace y un código CRC. El preámbulo es una frecuencia prefijada que sirve para sincronizar el receptor en frecuencia y estimar la duración de los símbolos. El código de acceso es único en caso de los canales de advertising, y en el caso de los canales de dato se trata de un código pseudoaleatorio generado por el dispositivo *initiator* al inicio de la conexión, y sirve para distinguir distintas comunicaciones que tengan lugar simultáneamente en el mismo canal radio. La figura 4.2 muestra un resumen del formato de los principales tipos de PDU.

La PDU de nivel de enlace puede a su vez ser de dos tipos diferentes con distinto formato, las PDU de los canales de *advertising* que sólo pueden transmitirse en dicho tipo de canales radio, y las PDU de los canales datos. Cada PDU consta de una cabecera de dos octetos, que es distinta en función del tipo de PDU, y un campo de datos de longitud variable (*payload*).

La cabeceras de las PDU de los canales de *advertising* incluyen un campo de 4 bits que indica el subtipo de PDU, un campo de 6 bits que indica la longitud del campo de datos y 2 bits adicionales *TxAdd* y *RxAdd* que aportan información sobre el tipo de direcciones utilizado.

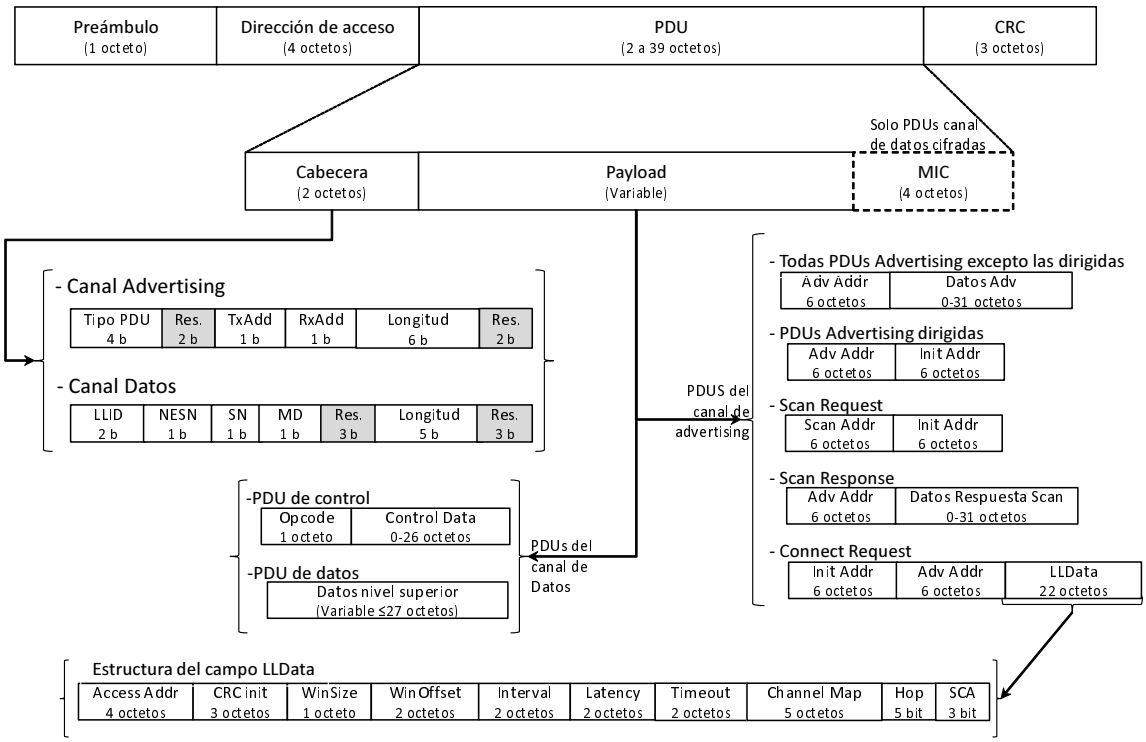


Figura 4.2: Formato de las tramas en *Bluetooth low energy*

Dentro de los subtipos de PDU del canal de *advertising* pueden distinguirse los siguientes:

- **PDUs de *advertising***

  Son las que se envían en los canales de advertising con el propósito de difundir información sobre el dispositivo que las transmite y hay 4 subtipos destinados a diferentes casos de uso:

  - *Advertising Indication PDU*, que se utiliza para indicar que el dispositivo que la emite acepta conexiones de cualquier dispositivo, y como ya se verá en la sección 4.2.3.2 el instante en el que esta PDU es recibida será utilizado por los dispositivos que quieran iniciar una conexión como referencia temporal para el establecimiento de la misma. Esta PDU lleva un campo con la dirección del dispositivo que la emite y unos datos opcionales.
  - *Advertising directed Indication* PDU, que se utiliza de forma similar a la anterior pero va dirigida a aceptar conexiones de un dispositivo concreto, por lo que además de la dirección del dispositivo que la emite, contiene la dirección del dispositivo al que va dirigida.
  - *Advertising non connectable Indication* PDU, cuya utilidad es la difusión de información por aquellos dispositivos que no pueden o no desean establecer conexiones, por lo que de nuevo además de la dirección del dispositivo que la emite la PDU contiene un campo de hasta 31 bytes de datos.

- *Advertising discoverable Indication* PDU, cuya principal utilidad es difundir activamente información para que el dispositivo que la emite pueda ser descubierto por otros dispositivos en su entorno, y de nuevo además de la dirección del dispositivo que la emite lleva un campo de datos de hasta 31 octetos que puede contener datos acerca del dispositivo.

- ***Scanning* PDUs**

  Estas PDUs se envían por los canales de *advertising* en el procedimiento de escaneo activo de dispositivos, que como se comentará en la sección 4.2.3.2 es uno de los mecanismos previstos por el estándar para encontrar dispositivos en la vecindad. En este caso hay dos subtipos de PDU, las de petición y las de respuesta:

  - Scanning request PDU, que es enviada por los dispositivos que se encuentran realizando un escaneo y contiene la dirección del dispositivo que realiza el escaneo y la dirección del dispositivo al que va dirigido.
  - Scanning response, que es enviada en respuesta a una PDU del tipo anterior cuando el dispositivo que la ha recibido estaba en estado de advertising. Este tipo de PDU contiene la dirección del dispositivo que la envía y datos

- ***Initiating* PDU**, que se utilizan en el procedimiento de establecimiento de conexión y son enviadas por el dispositivo que quiere iniciarla. Este tipo de PDU contiene tanto la dirección del dispositivo que inicia la conexión como la dirección del dispositivo con el que este se quiere conectar, y un conjunto de 22 octetos conteniendo datos y parámetros necesarios para el estabecimiento y configuración de la conexión, entre los que se encuentran principalmente parámetros que fijan la temporización (como $WinSize$, $WinOffset$, $Interval$, *latency* y *timeout* cuyo significado se comentará más adelante en la sección 4.2.3.3) y datos para establecer el mapa de canales válidos y la secuencia de saltos de frecuencia.

Las PDU del canal de datos también constan de una cabecera de 16 bits y un campo de longitud variable, y además pueden incluir un campo de comprobación de integridad del mensaje (MIC) para mejorar la seguridad de la comunicación. La cabecera de este tipo de PDU lleva un campo de 2 bits $LLID$ que indica el tipo de PDU, un campo de 5 bits que indica la longitud de la parte variable de datos de la PDU, dos campos de 1 bit para el número de secuencia de las PDUs enviadas y esperada, y un campo para indicar la existencia de más datos a transmitir en el dispositivo.

Las PDUs del canal de datos se transmiten por dichos canales una vez que se ha establecido una conexión entre dos dispositivos, y pueden utilizarse para enviar datos de usuario (paquetes de niveles superiores) o información de control y señalización del propio nivel de enlace, existiendo también diferentes subtipos de PDU según el caso de uso:

- **PDU de datos**. Las PDU de datos se utilizan para el transporte de datos de usuario, es decir, para el envío de información correspondiente a la capa superior de la pila de protocolo (L2CAP) entre dispositivos. La información contenida en las cabeceras de

estas PDU se puede utilizar también para el control de la comunicación, ya que por ejemplo un maestro puede enviar una PDU de este tipo vacía (sin datos) al esclavo símplemente para permitir que éste envíe datos en caso de tenerlos disponibles. Los diferentes campos de la cabecera permiten también indicar si un dispositivo tiene datos adicionales que enviar, si el mensaje enviado es el comienzo de un paquete L2CAP o una continuación de uno anterior, etc.

- **PDU de control**. Las PDU de control están destinadas al intercambio de información de señalización de nivel de enlace entre dos dispositivos, y constan de un campo de 1 octeto que indica el código de operación (o subtipo de PDU de control) y un campo de hasta 22 octectos con datos de control. Los distintos subtipos de PDUs de control se utilizan en diferentes casos de usos relacionados con el control del nivel de enlace, como por ejemplo la reconfiguración de los parámetros de conexión o del mapa de canales a utilizar en el salto de frecuencia, el cierre de la conexión, la puesta en marcha, detención y establecimiento de parámetros de encriptación y cifrado, la consulta de capacidades soportadas por el dispositivo remoto, etc.

4.2.3.2. Descubrimiento de dispositivos y formación de la red

Al igual que la tecnología *Bluetooth* original, la versión *low energy* define una serie de procedimientos para el descubrimiento de dispositivos vecinos y para el establecimiento de conexiones entre dispositivos para formar una red. Dichos procedimientos son equivalentes a los procedimientos de *Inquiry* y *Page* originales, pero se realizan de forma diferente y reciben un nombre distinto en *Bluetooth LE*. Para implementar estos procedimientos, también se definen una serie de estados que son aproximadamente equivalentes a los ya definidos por *Bluetooth* y previamente descritos en el capítulo 2.

Diagrama de estado de *Bluetooth low energy* La capa *link layer* define que un dispositivo *Bluetooth low energy* puede estar en uno de los siguientes estados (ver figura 4.3):

- *Standby*: Es el estado en el que está el dispositivo por defecto cuando no está formando parte de una red. En este estado el dispositivo no realiza transmisiones ni tiene la recepción activada.
- *Advertising*: En este estado el dispositivo transmite paquetes de anuncio en los canales de señalización y queda a la espera de recibir posibles respuestas de otros dispositivos disparadas por estos paquetes de anuncio, a las cuales a su vez responde. A este estado se entra desde el estado de *Standby*.
- *Scanning*: En este estado el dispositivo tendrá el receptor activado y configurado para recibir paquetes de anuncio de otros dispositivos enviados en alguno de los canales de señalización.
- *Initiating*: En este estado el dispositivo se encuentra escuchando en los canales de señalización los paquetes de anuncio de otros dispositivos y puede responder a algunos de ellos para iniciar una conexión.

- Connection: En este estado el dispositivo se encuentra sincronizado con otro, permitiendo la comunicación entre ellos. Uno de los dos dispositivos actuará como maestro y llevará el control del intercambio de datos, mientras que el otro estará actuando como esclavo.

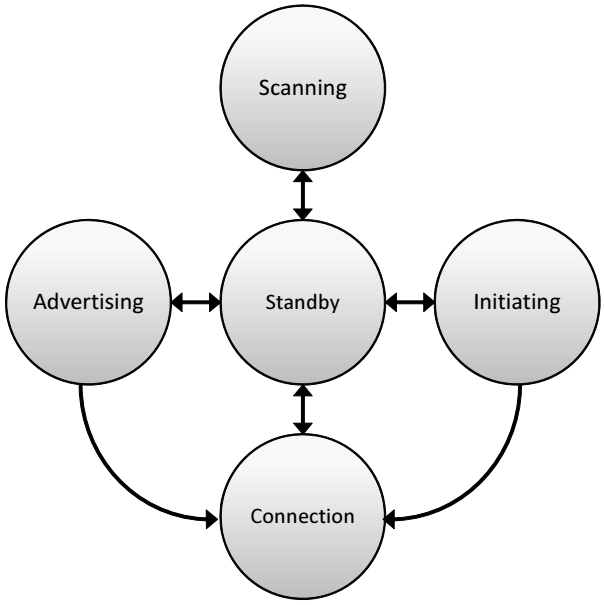


Figura 4.3: Diagrama de estados en *Bluetooth low energy*

Búsqueda y descubrimiento de dispositivos La detección de otros dispositivos compatibles en el entorno es necesaria para hacer posible la formación de redes espontáneas. Los procedimientos establecidos por *Bluetooth low energy* para este propósito hacen uso de la información difundida en los canales radio no destinados a conexión. Aquellos dispositivos que sean descubribles estarán en el estado de *advertising* y difundirán paquetes de *advertising* en los canales radio disponibles para ello, mientras que aquellos que estén buscando otros dispositivos en el entorno estarán en el estado de *scanning* y por tanto estarán escuchando en los canales de señalización los paquetes difundidos por otros dispositivos, y ocasionalmente podrán hacer uso de los paquetes de *scan request* para solicitar información adicional a los dispositivos encontrados (de nuevo utilizando los canales radio de señalización). En esta sección se describe el comportamiento de los dispositivos *advertiser* y *scanner* en el proceso de descubrimiento.

Dispositivo *advertiser* La temporización del envío de los diferentes tipos de PDU por un dispositivo en estado de *advertising* es determinada por los eventos de *advertising* que tienen lugar cuando el dispositivo es configurado en dicho estado por los niveles superiores. En cada evento de *advertising* se puede mandar una PDU por cada uno de los 3 canales radio de señalización, aunque el evento puede cerrarse sin haber llegado a enviar PDUs por todos los canales. El tiempo entre dos eventos de *advertising* $T_{advEvent}$, en cada uno de los cuales se pueden enviar PDUs por los canales de señalización, es la

suma de un término constante *advInterval* más un término pseudoaleatorio *advDelay*. El valor *advInterval* es un parámetro configurable por los niveles superiores que debe ser un múltiplo entero de 0,625ms en el rango de 20ms a 10,24s, mientras que *advDelay* es un valor pseudo aleatorio comprendido entre 0 y 10ms, que cambia para cada evento, y cuya finalidad es evitar que la temporización de los eventos de *advertising* sea completamente periódica, de forma que se eliminan sincronizaciones indeseables en los canales de señalización compartidos.

El procedimiento de *advertising*, como ya se ha mencionado, no sólo sirve para el descubrimiento de dispositivos, sino que también interviene en el establecimiento de conexión, e incluso puede utilizarse también para la difusión de información por dispositivos que no tengan capacidad de establecer conexiones. El comportamiento viene determinado por el tipo de PDU que se envíe por los canales radio de señalización en los eventos de advertising. En el caso del procedimiento de detección de dispositivos, las PDU que se utilizan son la *advertising indication* y la *advertising discoverable indication*. La primera de ellas también puede utilizarse en el establecimiento de conexión y por tanto es utilizada por dispositivos que son simultáneamente visibles y *conectables*, mientras que la segunda es utilizada por dispositivos que son visibles pero no aceptan conexiones.

En el caso de los dispositivos visibles y *conectables*, tras el envío de una PDU *advertising indication* por un canal radio de señalización, el dispositivo podría recibir a continuación y por el mismo canal una PDU de *scan request* o de *connection request* por parte de otro dispositivo que haya estado escuchando en dicho canal. La recepción de una PDU *connection request* indica que hay un dispositivo intentando conectarse, caso que se describirá en la siguiente sección. Si en cambio recibe una PDU *scan request*, es que hay un dispositivo cercano en estado de *scanning*, y se le deberá responder por el mismo canal con información adicional mediante una PDU *scan response*. Entre cada paquete intercambiado el estándar fija que debe haber un periodo de $150 \pm 2\mu s$, denominado IFS (*Inter Frame Spacing*) para permitir conmutar la radio de transmisión a recepción o viceversa. Este tiempo de espaciado IFS se aplica siempre que se intercambien varios paquetes en el mismo canal radio. Una vez intercambiadas las PDUs de *scan*, o si no se recibe ninguna, el dispositivo puede cambiar al siguiente canal radio de señalización y enviar una nueva PDU de *Advertising Indication*, repitiendo el proceso hasta recorrer los 3 canales de señalización o hasta que finalice el evento de *advertising*. En cualquier caso, entre el comienzo del envío de una PDU de *advertising* por un canal y el de la siguiente por otro no deben transcurrir más de 10 ms. La temporización del proceso se muestra en la figura 4.4

Si el dispositivo es visible pero no admite conexiones, el procedimiento y la temporización son iguales a los descritos para el caso anterior, pero se emplea la PDU *Advertising Discoverable Indication*, y tras el envío de estas sólo se admite la recepción de PDUs de *scan request*, siendo rechazadas como incorrectas las PDU de *connection request*.

La temporización descrita anteriormente se utiliza en todos los procedimientos de *advertising* no dirigidos. Existe un procedimiento de *advertising* dirigido, que sólo se utiliza en el establecimiento de conexión, y que hace un uso más agresivo de los canales. En este procedimiento se envían contínuamente PDUs de *Advertising directed Indication* con un espaciado de menos de 3,75 ms entre PDUs enviadas por el mismo canal siempre que no

se reciba respuesta (Ver figura 4.5). Este procedimiento hace un uso bastante intensivo de los canales de *advertising* y conlleva un gasto energético muy considerable, por lo que sólo debe utilizarse en el establecimiento rápido de conexiones.

Dispositivo Scanner Durante el estado *scanning*, en el que se entra bajo control de los niveles superiores de la pila de protocolo, el dispositivo permanece a la escucha en los canales radio de señalización a la espera de recibir PDUs de *advertising*, y por cada PDU no duplicada recibida debe enviar un informe a la capa de nivel superior.

La temporización de la escucha realizada por el *scanner* es regulada por dos parámetros que son configurados por el nivel superior al iniciar el proceso de *scanning*, $scanWindow$ y $scanInterval$. El parámetro $scanWindow$ fija el intervalo de tiempo durante el cual el dispositivo *scanner* permanece a la escucha en un determinado canal radio, mientras que $scanInterval$ fija la periodicidad con la que se repite la escucha. Ambos parámetros deben ser múltiplos de 0,625ms, deben estar comprendidos entre 2,5 ms y 10,24s, y obviamente $scanInterval$ siempre debe ser mayor o igual que $scanWindow$. El dispositivo *scanner* sintonizará de forma consecutiva los canales radio de señalización que hayan sido configurados por el nivel superior, utilizando uno distinto por cada intervalo de escucha y pasando alternativamente por todos los canales habilitados.

Según se configure por el protocolo de nivel superior, el procedimiento de *scanning* puede ser de tipo pasivo o activo. Si es pasivo, entonces el dispositivo *scanner* únicamente se limita a escuchar en los canales radio de señalización y mandar informes al nivel superior con información de los *advertisers* detectados. Si el *scanning* es activo, entonces el dispositivo scanner puede solicitar información adicional a los dispositivos encontrados cuando la PDU recibida es de tipo *Advertising indication* o *Advertising discoverable indication*. Para ello, tras recibir la PDU del dispositivo advertiser, envía por el mismo canal radio una PDU de *scan request* hacia el dispositivo *advertiser*, el cual debe responder con una PDU de *scan response*. Como ya se ha mencionado anteriormente, entre la recepción de una PDU y el envío de otra por el mismo canal se debe dejar un tiempo de guarda IFS para permitir la conmutación de la radio de transmisión a recepción.

En el *scanning* activo el nivel de enlace puede enviar varias PDUs de *scan request* consecutivas al mismo dispositivo y también puede intercalar envíos de PDUs de este tipo a diferentes dispositivos. Puesto que el acceso a los canales radio de señalización es compartido, se debe implementar un procedimiento de *backoff* para reducir colisiones en la medida de lo posible y evitar que estas ocurran de forma repetida. Por ello, se implementan dos variables $upperLimit$ y $backoffCount$ que se inicializan a 1 al comenzar el proceso de *scanning*. $backoffCount$ se decrementa cada vez que se recibe una PDU *Advertising indication* o *Advertising discoverable indication*, y cuando llega a 0 el *scanner* puede proceder a enviar la PDU de *scan request*. Si se recibe la PDU de *scan response*, la comunicación se considerará como exitosa, y si no, se considerará un fallo. Por cada dos fallos, se duplica la variable $upperLimit$ y por cada dos aciertos se divide a la mitad (hasta llegar a 1). Para cada nueva PDU de *scan request* que se quiera transmitir $backoffCount$ se fija aleatoriamente a un valor entre 1 y $upperLimit$.

Establecimiento de la conexión El procedimiento de establecimiento de conexión permite que dos dispositivos se unan formando una piconet, de la cual uno será el maestro y otro el esclavo. Durante este procedimiento los dispositivos deben sincronizarse y ponerse de acuerdo en los parámetros de temporización que van a utilizar, para que durante la conexión puedan comunicarse de forma coordinada. Para ello, uno de los dispositivos debe estar en el estado *advertising* aceptando conexiones, mientras que el otro debe entrar en el estado *initiating*. El dispositivo *initiator* permanecerá en un primer momento escuchando en los canales radio de señalización a la espera de que el dispositivo con el que quiere conectar difunda una PDU. Una vez detectada dicha PDU, intentará conectar con él por el mismo canal radio. Al finalizar el proceso el dispositivo *initiator* quedará como maestro y el dispositivo *advertiser* como esclavo. La temporización de la fase inicial del establecimiento de conexión se muestra en la figura 4.6

Dispositivo advertiser En este escenario el comportamiento y la temporización del dispositivo *advertiser* es similar al ya comentado para el escenario de descubrimiento de dispositivos. En este caso, el *advertiser* puede utilizar la PDU de *advertising indication* que también se podía utilizar en en el descubrimiento, o bien puede utilizar la PDU *Advertising directed PDU* si quiere indicar que acepta conexiones sólo de un dispositivo. Tanto en un caso como en el otro, tras el envío de la PDU por el canal radio de señalización, el dispositivo *advertiser* permanece atento a la recepción por el mismo canal de una PDU de *connection request* conteniendo su dirección. Si se recibe este tipo de PDU enviada por un dispositivo vecino, el dispositivo finaliza el evento de *scanning* y pasa del estado *scanning* al estado *connection*.

Dispositivo initiator El dispositivo que inicia el establecimiento de la conexión entra en el estado *initiating* cuando se lo solicite el protocolo de nivel superior. En este estado el dispositivo comienza por escuchar en los canales radio de señalización hasta detectar una PDU del dispositivo con el cual se quiere establecer la conexión. El comportamiento y la temporización en esta etapa son muy similares a las de un dispositivo *scanner*, de forma que se escucha durante un intervalo temporal en cada uno de los canales radio habilitados y este intervalo de escucha se repite cada cierto tiempo, recorriendo consecutivamente los diferentes canales habilitados. De nuevo los parámetros $scanWindow$ y $scanInterval$ son fijados por los niveles superiores de la pila de protocolos, teniendo el mismo rango y limitaciones ya mencionadas para el dispositivo *scanner*. La diferencia con el procedimiento anterior radica en que se ignoran las PDUs *Advertising discoverable indication* y en cambio sí que se procesan las PDU Advertising directed indication si la dirección indicada coincide con la del dispositivo *initiator*. Además, al recibir las PDUs esperadas, se envía hacia el dispositivo *advertiser* una PDU de tipo *connection request*, tras lo cual el dispositivo *initiator* pasa al estado *connection*, cuyo funcionamiento se describe en la siguiente sección.

4.2.3.3. Comunicación entre dispositivos conectados

El nivel de enlace de un dispositivo *Bluetooth low energy* entra en el estado *connection* cuando como *initiator* completa el envío de una PDU *connection request* o cuando como *advertiser* recibe y acepta una PDU de *connection request* de un *initiator*. En este punto la conexión se considera creada, pero no se considera aún establecida, y no se considerará así hasta que los dispositivos se intercambien correctamente las primeras PDUs del canal de datos. La diferencia entre conexión creada y establecida radica en el valor del parámetro *supervision timeout* que se emplea.

Temporización de la conexión Como ya se ha mencionado anteriormente, cuando dos dispositivos se encuentran participando en una conexión uno de ellos tiene el papel de maestro, mientras que el otro toma el papel de esclavo. El nivel de enlace del dispositivo maestro es el que controla la temporización de los eventos de conexión, que son los puntos de sincronización entre maestro y esclavo.

La temporización de los eventos de conexión viene determinada por los parámetros *connInterval* y *connSlaveLatency*. El instante de comienzo de un evento de conexión se denomina punto de anclaje, y se repite de forma periódica con un espaciado regular determinado por *connInterval*. Al comienzo de un evento de conexión el maestro debe comenzar a transmitir una PDU del canal de datos al esclavo. Durante un evento de conexión los dispositivos maestro y esclavo se intercambiarán PDUs del canal de datos de forma alternativa, y mientras ambos dispositivos continúen enviando paquetes el evento de conexión se considerará abierto. Una vez cerrado no podrán volverse a intercambiar PDUs hasta el siguiente evento de conexión. Los eventos de conexión no deben superponerse, por lo que el maestro deberá garantizar que un evento de conexión finalice al menos un IFS antes del punto de anclaje del siguiente evento de conexión. Un ejemplo de temporización de la conexión en *Bluetooth low energy* se muestra en la figura 4.6.

Durante todo el evento de conexión las PDU de canal de datos se intercambiaran por el mismo canal radio, cuyo índice será determinado por el maestro y el esclavo. En el siguiente evento de conexión se elegirá un canal radio de datos con un índice distinto, siguiendo una secuencia de saltos pseudoaleatoria que ambos dispositivos conocen, de forma que se consiga la diversidad por salto de frecuencia.

El parámetro *connInterval* es fijado por los niveles superiores del dispositivo *initiator* y comunicado al dispositivo que acepta la conexión en el establecimiento de la misma y debe ser un múltiplo de 1,25ms comprendido entre 7,5ms y 4.0 s. El parámetro *connSlaveLatency* permite reducir el número de eventos de conexión a los que el dispositivo esclavo debe atender, y define el número máximo de eventos de conexión consecutivos en los que puede permanecer en estado latente sin escuchar al maestro. El valor de este parámetro debe ser tal que no produzca un *timeout* de la conexión, por lo que su valor de debe de poner conforme al valor $connSupervisionTimeout$ establecido. Cuando un dispositivo esclavo no recibe un paquete del maestro en un punto de anclaje en el que lo espera debe dejar de aplicar el mecanismo de latencia (es decir, debe escuchar en todos los puntos de anclaje) hasta volver a recibir un paquete del maestro.

Los dispositivos maestro y esclavo deben vigilar continuamente el estado de la conexión,

ya que ésta puede verse interrumpida sin previo aviso debido a diferentes causas como la aparición de interferencias radio, el movimiento de un dispositivos fuera del alcance del otro o la pérdida de alimentación de uno de ellos. Para ello se establece un temporizador de supervisión, que es incrementado con el tiempo y reiniciado cada vez que se recibe un paquete válido del otro dispositivo. Si este temporizador llega a un determinado límite, el nivel de enlace del dispositivo dará la conexión por perdida e informará a los niveles superiores. El límite fijado depende de si la conexión está creada o establecida. En el primer caso, si el temporizador de supervisión supera $6 \cdot connInterval$ la conexión se dará por fallida. Si la conexión ya estaba establecida, entonces el límite utilizado es un parámetro denominado *connSupervisioTimeout* que es fijado en el establecimiento de la conexión por el dispositivo *initiator* según lo especificado por los niveles superiores. Este parámetro debe ser un múltiplo de 10ms en el rango de 100ms a 32s y lógicamente debe ser superior a $(1 + connSlaveLatency) \cdot connInterval$ para evitar que el estado latente del esclavo provoque un timeout de supervisión en el maestro. Si una conexión establecida se supera el límite del temporizador de supervisión, no se volverá a transmitir ninguna PDU, el nivel de enlace abandonará el estado *connection* pasando al estado *standby*, y se notificará a las capas superiores la pérdida de la conexión.

Finalización del establecimiento de la conexión Como ya se ha mencionado, la temporización de la conexión *Bluetooth low energy* una vez establecida viene determinada por los eventos de conexión, que se repiten de forma regular a partir del primer punto de anclaje. Éste primer punto de anclaje viene determinado por la primera transmisión de una PDU desde el maestro hacia el esclavo utilizando el canal de datos tras la creación de la conexión, momento a partir del cual la conexión se considera establecida.

Para dotar al maestro de una mayor flexibilidad a la hora de establecer varias conexiones, el estándar establece que el maestro tiene la opción de elegir el instante de tiempo del primer punto de anclaje dentro de un margen, lo que le permite determinar el *offset* temporal de la conexión. Para determinar los márgenes en los cuales debe tener lugar el primer punto de anclaje, la capa de enlace del *initiator*/maestro establece dos parámetros que son comunicados al dispositivo *advertiser*/esclavo en la PDU de *connection request*: $transmitWindowOffset$ y $transmitWindowSize$. Ambos deben ser múltiplos enteros de 1,25ms, estando primero en el rango de 0 a *connInterval* y el segundo de 10ms a $(connInterval - 1,25ms)$. El punto de anclaje tendrá lugar dentro de una ventana temporal que comienza $transmitWindowOffset + 1,25ms$ después del fin de la transmisión de la PDU de *connection request* y cuya duración es de $transmitWindowSize$. La temporización de la fase final de la conexión se muestra también en la figura 4.6. A continuación se describe de forma algo más detallada el comportamiento del dispositivo *initiator/maestro* y del dispositivo esclavo mientras se completa el establecimiento de conexión.

Comportamiento del maestro Tras el envío de la PDU *connection request* el *initiator* cambia al estado *connection* y pasa a comportarse como maestro. El valor del temporizador de supervisión se inicializa a 0 y su límite se establece en $6 \cdot connInterval$. El dispositivo debe comenzar a transmitir la primera PDU por el canal de datos dentro

de la ventana definida y comunicada previamente al esclavo (sólo el comienzo del paquete debe estar en la ventana, su transmisión se puede completar fuera de ésta). El comienzo de la transmisión determina el primer punto de anclaje y por tanto la temporización de todos los demás eventos de la conexión, ya que cada uno tendrá lugar *connInterval* después del anterior. Si tras el envío de la primera PDU se recibe respuesta por parte del esclavo, el maestro puede transmitir un nuevo paquete hacia el esclavo. Si no se recibe, el evento de conexión se considera cerrado y el maestro volverá a intentar enviar información al esclavo en el siguiente evento, utilizando para ello el siguiente canal radio de satos que determine la secuencia de saltos.

Comportamiento del esclavo Tras la recepción de una PDU de *connection request* que puede aceptar, el *advertiser* cambia al estado *connection* y pasa a comportarse como esclavo, inicializando el valor del temporizador de supervisión a 0 y fijando su límite en $6 \cdot connInterval$. El dispositivo debe activar la recepción durante toda la ventana de transmisión indicada configurada por el *initiator/maestro* y quedar a la espera de recibir la primera PDU procedente de éste por el canal de datos. La primera PDU recibida cuyo código de acceso sea el correcto, independientemente de si el CRC lo es o no, determina el primer punto de anclaje de la conexión. Si no se recibe ninguna PDU, el esclavo debe volver a activar la recepción en una segunda ventana que comenzará lugar un intervalo *connInterval* después de la anterior y tendrá la misma duración. En esta segunda ventana el receptor escucha en un canal radio de datos distinto determinado por el algoritmo que controla la secuencia de saltos, y el contador de eventos de conexión se incrementa aunque no se haya completado el establecimiento de conexión.

Mantenimiento de la sincronización maestro-esclavo El estandar exije una exactitud mínima de $\pm 50 ppm$ para el reloj de los dispositivos en modo activo y de $\pm 500 ppm$ cuando están dormidos (parámetro denominado SCA, *Sleep Clock Accuracy*). La primera se aplica a la temporización de envío de las PDU mientras el evento de conexión está abierto (y por tanto fija la tolerancia de los IFS), mientras que el segundo se aplica al tiempo que transcurre desde la finalización de un evento de conexión hasta el comienzo del siguiente, y es más relajada para permitir una mayor reducción del consumo en esta fase de inactividad.

La diferencia de frecuencia entre los relojes del maestro y del esclavo introduce una cierta deriva en la temporización, por lo que la norma establece que el esclavo debe resincronizarse con el maestro en cada nuevo punto de anclaje en el que deba estar activo. Esto se realizará independientemente de si la PDU recibida contiene un CRC correcto o no. El esclavo debe estimar cuándo va a ocurrir cada punto de anclaje y despertarse y activar la radio algo antes del mismo y no apagarla hasta transcurrido un cierto tiempo si no recibe nada. La ventana de escucha dependerá de la exactitud real del SCA del maestro (que es comunicada en la PDU de *connection request*, y debe hacerse mayor conforma más tiempo ha transcurrido desde el último punto de anclaje según la siguiente ecuación (asumiendo $masterSCA$ y $slaveSCA$ en ppm):

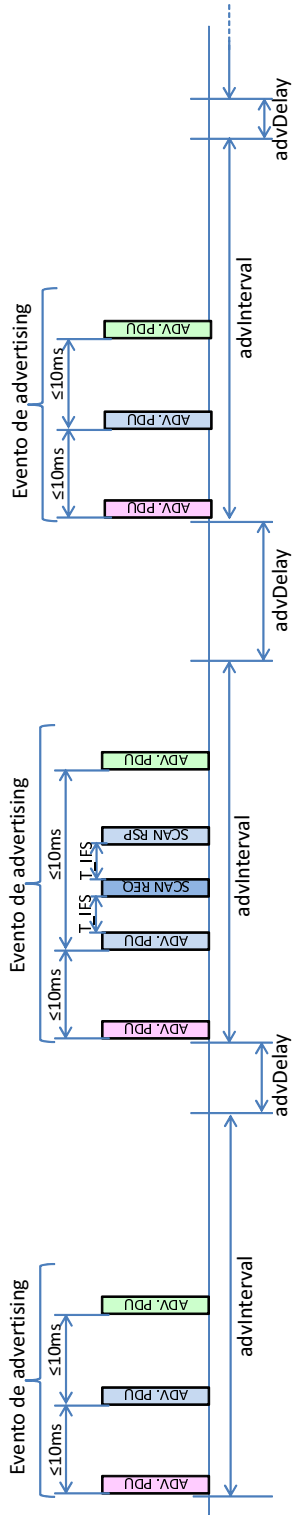


Figura 4.4: Temporización de los canales de *advertising*

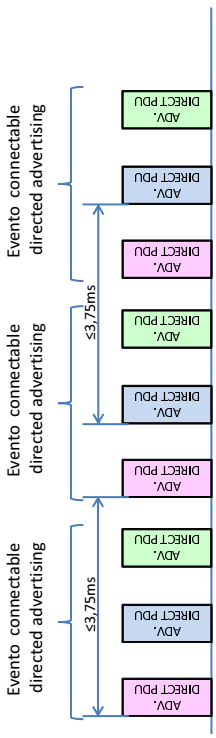


Figura 4.5: Temporización del procedimiento de *advertising* dirigido

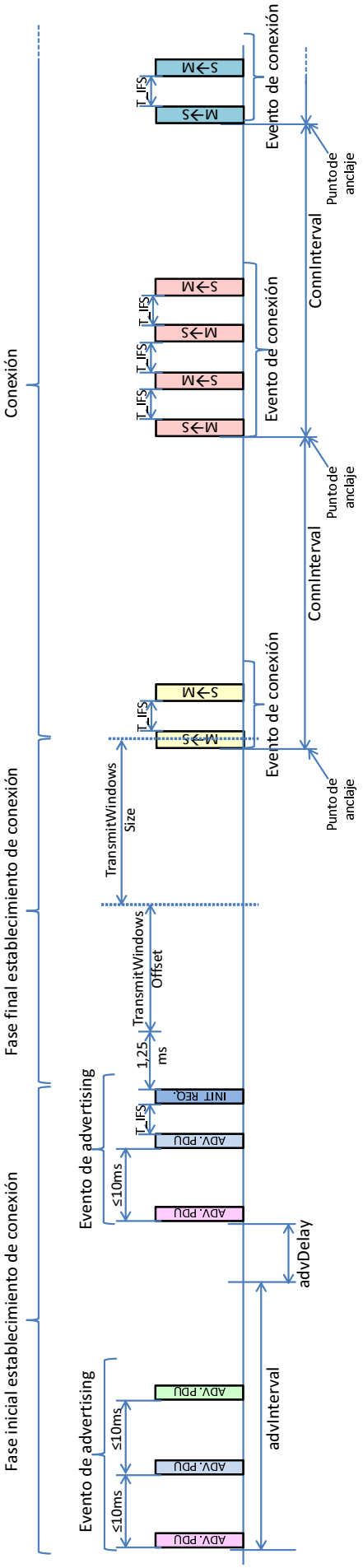


Figura 4.6: Temporización del establecimiento y de la conexión

$$windowWidening = \frac{(masterSCA + slaveSCA) \cdot timeSinceLastAnchor}{1000000} \quad (4.1)$$

Este parámetro $windowWidening$ define el tiempo que el receptor del esclavo debe estar activo antes y después del punto de anclaje estimado y debe ser menor que $(connInterval/2) - T_{IFS}$, dándose por perdida la conexión si se llega a superar.

Este parámetro también se aplica en el caso del establecimiento de conexión como ensanchamiento de la ventana configurada en el establecimiento de conexión para las subsiguientes escuchas cuando no se reciben PDUs del maestro.

4.2.3.4. Control de enlace

La capa de control de enlace se encarga de establecer y negociar determinados aspectos del funcionamiento de la conexión entre dos niveles de enlace, y básicamente incluye procedimientos para el cambio de los parámetros de la conexión y para la puesta en marcha de la encriptación de datos.

4.2.4. Protocolo ATT y perfil GATT

Al igual que *Bluetooth* clásico, la variante *low energy* implementa diferentes perfiles de aplicación definidos por la norma para distintos casos de uso y servicios. Para soportar estos perfiles de una forma más sencilla y ligera, la especificación 4.0 incluyó un nuevo protocolo de gestión de atributos, denominado ATT. El protocolo de gestión de atributos ATT está específicamente pensado para implementar los perfiles definidos para *Bluetooth LE*, aunque al soportarse sobre L2CAP puede ser utilizado también con el controlador *Bluetooth BR/EDR* y por tanto ser integrado en dispositivos antiguos. El protocolo ATT sigue una arquitectura cliente/servidor, en la que un dispositivo que actúa como servidor ofrece una serie de atributos que pueden ser leídos o modificados por el que actúa como cliente. Además, el protocolo permite que el cliente puede suscribirse a algunos atributos, y que el servidor le envíe asíncronamente mensajes de notificación cuando los atributos a los que se ha suscrito cambien. No todos los atributos se pueden escribir o leer, por lo que el protocolo también soporta la gestión de permisos asociados a los atributos, incluyendo permisos relativos a la seguridad o encriptación de los datos, ya que hay atributos que no se pueden intercambiar si antes no se ha establecido una conexión segura. Todos los atributos son arrays de octetos de longitud variable, cuya codificación y tamaño depende del tipo de atributo que es identificado mediante un código UUID (*Universally Unique Identifier*) definido por el *Bluetooth SIG*. Cada atributo del servidor tiene además un *manejador*, que es un número de 16 bits asignado internamente que lo identifica y permite referenciarlo. El protocolo establece mecanismos para poder buscar atributos, es decir, encontrar el *manejador* de un atributo de un determinado tipo (o de los manejadores de un grupo de atributos de un determinado tipo).

El perfil GATT, especificado por la versión 4.0 del *Core* de las especificaciones define cómo utilizar el protocolo ATT para descubrir los servicios y características ofrecidos por un dispositivo *Bluetooth*, lo que se realiza a través de descriptores de servicio codificados a su vez en forma de atributos. Los servicios, características y atributos que deben implementar los dispositivos de un determinado perfil se definen por el *Bluetooth SIG* en documentos separados.

4.3. Trabajos relacionados

Bluetooth low energy es la tecnología de área personal que ha experimentado una de las más rápidas expansiones en los últimos años y que actualmente se posiciona como uno de los pilares fundamentales que van a dar soporte a la internet de la cosas [Dec15; RMHV15]. Gracias al desarrollo de módulos radio *Bluetooth* duales capaces de operar en los modos BR/EDR tradicional y *low energy*, y a la progresiva incorporación y adaptación de los protocolos de nivel *host* en los sistemas operativos más utilizados, actualmente se encuentra disponible en millones de terminales de comunicación personal como portátiles, tabletas y *Smartphones*. Gracias a esta disponibilidad y a la idoneidad de su diseño *Bluetooth low energy* está siendo adoptada como tecnología de interconexión por diferentes dispositivos periféricos alimentados a batería, con particular éxito en el campo de los dispositivos médicos [WSL13] y vestibles (*wereables*) [Wei14; Kir14], así como en aplicaciones de hogar inteligente [Dec14; CP15] e incluso en aplicaciones para redes vehiculares [LTT15; BFCE15].

Es por estas grandes expectativas que desde la adopción de las especificaciones *low energy* por el *Bluetooth SIG* diversos investigadores han centrado su atención en el estudio de las prestaciones y rendimiento de la tecnología, si bien el número de estudios existentes es menos elevado que para las otras tecnologías evaluadas en los capítulos anteriores de esta Tesis, y se centran más específicamente en la evaluación del consumo. Además, en comparación con con los estudios publicados en la literatura para otras tecnologías como *Bluetooth*, cabe destacar un aumento de la utilización de una metodología experimental basada en la realización de pruebas con dispositivos reales para obtener y contrastar los resultados. El retardo y el *througput* son otros de los dos parámetros principalmente considerados junto con el consumo.

En uno de los primeros estudios publicados sobre la tecnología [GDP11] los autores proponen un modelo analítico para estimar el *througput* máximo en función del parámetro *connection interval* para diferentes tasas de error de bit. Los resultados se contrastan mediante simulaciones, aunque no se aporta información sobre la plataforma de simulación utilizada. Este estudio es ampliado por los mismos autores en [GOP12], incluyendo un estudio analítico del consumo de los dispositivos esclavos al mantener una comunicación punto a punto, así como un estudio mediante simulación de la latencia de la comunicación. En este trabajo también se incluyen algunas pruebas realizadas en entornos reales utilizando dispositivos basados en el módulo CC2540 de Texas Instruments [Cc2b], poniendo de manifiesto que el *throughput* esperado puede degradarse notablemente por limitaciones no impuestas por el estándar sino por las implementacione comerciales, como por ejemplo el

número máximo de paquetes intercambiados en un *connection event*. El modelo utilizado en este trabajo para el estudio teórico del consumo está basado en el modelo propuesto por una nota de aplicación del propio fabricante del dispositivo [KL11], que permite estimar el consumo en función de parámetros como el *connection interval* y en la que también se basan otros estudios posteriores.

Puesto que el consumo es uno de los principales elementos diferenciadores con los que *Bluetooth low energy* hace frente a sus competidores, diversos trabajos abordan el estudio del consumo desde un enfoque comparativo. Así por ejemplo, los autores de [SHNN12] realizan una comparativa entre el consumo de *Bluetooth low energy* y *ZigBee* mediante el análisis de módulos reales basados en los dispositivos CC2540 y CC2530 de Texas Instruments respectivamente. Aunque interesante, el estudio presenta algunas limitaciones, como por ejemplo que no se tiene en cuenta el efecto del CSMA/CA en el consumo de *ZigBee* y que la resolución temporal de las medidas realizadas (cuya metodología además no se indica) no es muy buena. En [DHTS13] se realiza una comparativa entre los consumos de sensores basados en implementaciones comerciales de *ZigBee*, *ANT+* y *Bluetooth low energy* para diferentes tiempos entre envío de datos. Para la medida del consumo se utiliza el circuito INA226, que integra un amplificador de transconductancia (similar al INA196) con un conversor A/D controlado mediante un interfaz serie. Por último en [MPT13] se utilizan métodos analíticos complementados con algunas pruebas en escenarios reales para realizar una comparativa más amplia en términos no sólo de consumo sino también de *throughput* entre *Bluetooth low energy*, *802.15.4* y SimpliciTI (un protocolo propietario de Texas Instruments [Sim]). En [AY15] se realiza una profusa revisión de diferentes posibles mecanismos y propuestas realizadas en la literatura para mejorar el consumo energético en diferentes tecnologías WPAN, incluyendo *Bluetooth LE*, IEEE 802.11.ah y *Z-Wave*. El trabajo [MPJ13] presenta un análisis cualitativo basado en el estudio de diversas notas de aplicación y hojas de características de dispositivos comerciales y realiza una comparativa entre *ZigBee*, *Bluetooth LE* y *WiseMAC* enfocada hacia redes de área corporal para aplicaciones de monitorización médica y deportiva.

Los estudios mencionados hasta ahora se centran principalmente en el análisis de las prestaciones y consumo de los dispositivos una vez establecida la comunicación (es decir, asumiendo que permanecen todo el tiempo en modo conectado). Las prestaciones y consumo energético durante la fase de conexión son estudiados también por algunos autores, puesto que pueden tener importancia en función de la aplicación o en el caso de que debido a interferencias se corte la conexión y sea necesario restablecerla. Por ejemplo, en [LCMX13] se realiza un estudio analítico sobre el consumo del dispositivo *advertiser* (que finaliza como esclavo) durante el establecimiento de conexión, en función de los parámetros configurados para la misma. En este análisis se utilizan los valores de consumo especificados por Texas Instruments para el dispositivo CC2541 en su hoja de características, así como un modelo analítico de la duración de las diferentes fases del procedimiento de conexión derivado por los mismos autores en [LCM12b; LCM12a]. Por otra parte, el trabajo [Mik14a] analiza el problema de las posibles colisiones durante la fase de conexión cuando varios dispositivos vecinos están simultáneamente intentando realizar dicho procedimiento y por tanto transmitiendo en los canales de *advertising*, y plantea un mecanismo compa-

tible con el estándar para optimizar el establecimiento de conexión. El estudio es llevado a cabo mediante simulación utilizando un modelo basado en el *framework* MiXim para OMNET++ desarrollado por el mismo autor y descrito en [Mik14b]. Este mismo modelo de simulación es también utilizado en [MH15] para proponer diversos mecanismos para mejorar el *throughput* y la eficiencia energética en entornos con varios nodos. La latencia del proceso de descubrimiento de dispositivos en diversos escenarios con diferente número de nodos *advertisers* y *scanners* es también estudiado en [CPH+14] mediante un complejo modelo matemático validado mediante simulaciones con una herramienta propietaria que no se detalla. Un estudio similar, pero realizado mediante la combinación de simulaciones con experimentos reales es presentado en [TSPB15] con objeto de evaluar el impacto de la interferencia mutua en la latencia y el consumo de los procedimientos de búsqueda y establecimiento de la conexión.

Puesto que *Bluetooth low energy* se ha revelado como una de las tecnologías que pueden ser claves en el desarrollo de la IoT, algunos autores [WXLC13; NGI+14] e incluso organizaciones como el IETF [NSI+15] se han interesado por el establecimiento de mecanismos para facilitar la integración de IPv6 sobre *Bluetooth LE* siguiendo un enfoque similar al de 6LoWPAN. En paralelo, estas demandas han llevado al *Bluetooth SIG* a especificar el perfil IPSP [SKB+14] que fué hecho público a finales de 2014 como parte de la revisión 4.2 del estándar, que también introduce otras modificaciones en la especificación de los protocolos base para mejorar el soporte para IPv6 (como la introducción de paquetes con un mayor *payload*). A pesar de estas iniciativas la topología de red soportada sigue siendo en estrella, aunque están empezando a aparecer algunas propuestas para permitir la formación de redes malladas [KLJ15] y el *Bluetooth SIG* ha creado recientemente un grupo de trabajo para su estandarización [RMHV15]. Algunos estudios preliminares han evaluado también la posibilidad de implementar comunicaciones multisalto utilizando los mecanismos actualmente soportados por la tecnología [MT13], pero los resultados obtenidos en términos de retardo y consumo no son buenos.

4.4. Aportaciones al estudio y modelado del consumo

4.4.1. Sistema de prueba

El sistema de prueba utilizado para obtener los resultados incluidos en esta sección se muestra en la figura 4.7, y consta de dos nodos *Bluetooth low energy* basados en sistemas de desarrollo para el dispositivo CC2541 de Texas Instruments [Cc2c]. Este dispositivo es un SoC (*System on Chip*) que integra una CPU basada en la arquitectura 8051 de Intel junto con el transveiver radio *Bluetooth LE* y algunos periféricos que ayudan a gestionar la temporización requerida por el nivel *link-layer* para mantener la sincronización entre los dispositivos que intervienen en la comunicación. Para implementar el resto de niveles de la pila de protocolos *Bluetooth low energy* y la aplicación, el dispositivo dispone de 8 kB de memoria RAM y 256 kB de memoria Flash. La implementación de la pila de protocolo completa para esta plataforma, junto con algunas aplicaciones de ejemplo, es proporcionada por el propio Texas Instruments como parte del sistema de desarrollo [Ble].

En el sistema de pruebas utilizado, uno de los nodos es configurado con una aplicación que emula un dispositivo sensor que ofrece un servicio de envío de datos al que otros dispositivos pueden suscribirse. El otro nodo es programado con una aplicación de test que permite controlar su funcionamiento desde un PC para realizar diferentes operaciones como búsquedas de dispositivos, establecimiento de conexiones y configuración de sus parámetros. Este nodo es utilizado para establecer la conexión con el dispositivo sensor y suscribirse al servicio de envío de datos, comportándose como maestro de la conexión *Bluetooth low energy*.

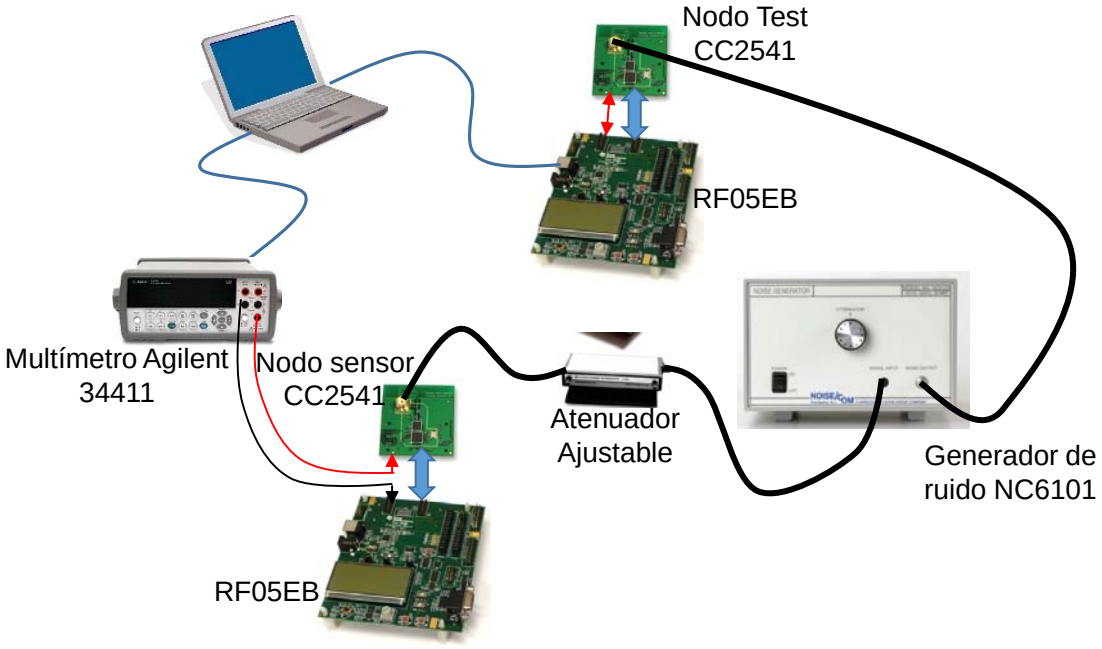


Figura 4.7: Sistema de pruebas para medir el consumo en *Bluetooth low energy*.

El nodo cuyo consumo se caracteriza es el dispositivo que se comporta como sensor. El *firmware* de este dispositivo se ha programado cuidadosamente de forma que aquellos pines del SoC no utilizados se configuren como salidas para evitar fugas de corrientes que distorsionen el consumo del dispositivo. El *firmware* original se ha modificado de forma que tras el cierre o el corte de una conexión el dispositivo activa indefinidamente el *advertising* hasta que se establezca una nueva conexión.

Al igual que en la sección 3.4.1, para estimar la corriente media consumida por el nodo sensor se utiliza de nuevo un multímetro 34411 de Agilent (actualmente Keysight), que puede ser también controlado desde un PC. Como ya se comentó en el capítulo 3, este multímetro ofrece diferentes modos de funcionamiento, que permiten tanto la medida del consumo instantáneo (con una resolución de hasta 50 kS/s) como la estimación del consumo medio en intervalos de tiempo largos (mediante el promediado de los valores obtenidos por el conversor de integración contínua). Para facilitar la realización automática de las pruebas se ha desarrollado una aplicación para el entorno de programación LabView que, además de permitir configurar y controlar la operación del multímetro y leer los datos adquiridos desde el PC, controla la operación y parámetros de funcionamiento del enlace

Bluetooth LE a través del nodo de test.

Para evitar interferencias no controladas, los terminales RF de los nodos CC2541 que intervienen en la comunicación se conectan directamente mediante cable coaxial con conector SMA en lugar de utilizar antenas. Para comprobar el efecto de la influencia de posibles pérdidas o interferencias en el consumo, éstas se emulan de forma controlada mediante un atenuador ajustable y un generador de ruido AWGN.

4.4.2. Caracterización y modelo del consumo

Al igual que en los capítulos anteriores, para caracterizar el consumo se analiza en primer lugar la forma de onda del consumo de corriente configurando el multímetro con la máxima resolución temporal posible. La figura 4.8 muestra por ejemplo el comportamiento de la corriente cuando el dispositivo está participando en una conexión como esclavo, para un intervalo de conexión de 12,5 ms (parámetro $connInterval = 10$) y sin envío de datos de aplicación. En la figura se observa el claro comportamiento periódico de la actividad necesaria para el mantenimiento de la conexión. El consumo del nodo durante los periodos de inactividad puede estimarse con mejor resolución utilizando la configuración del multímetro con periodos de integración altos y es de unos $30\mu A$.

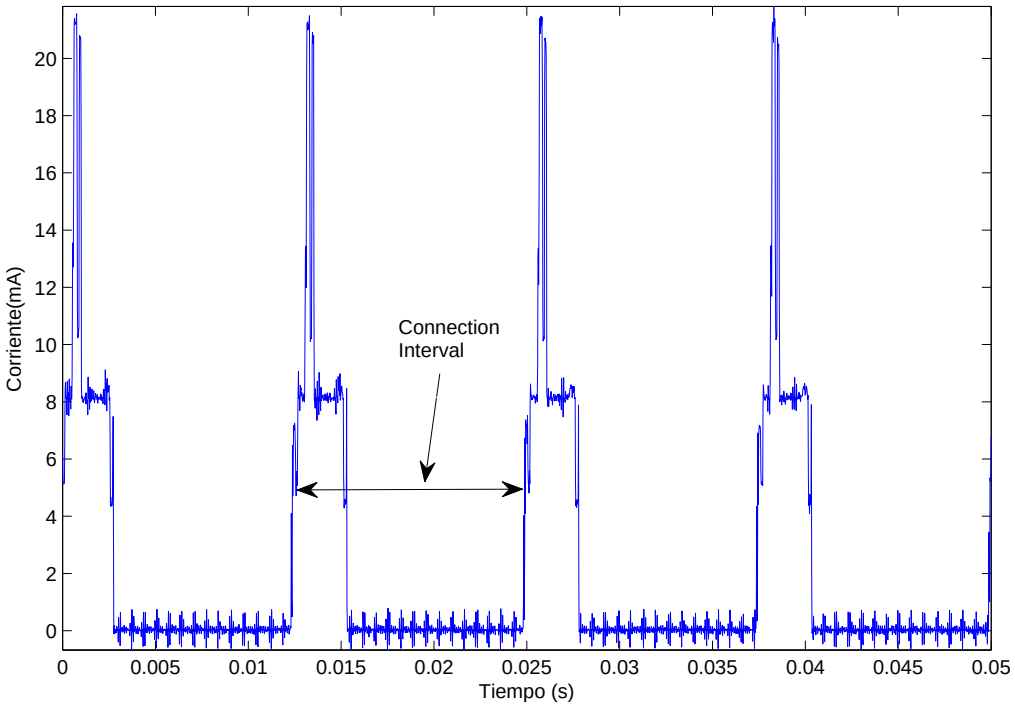


Figura 4.8: Forma de onda de la corriente consumida por el dispositivo CC2541 para un intervalo de conexión de 12,5 ms

La figura 4.9 muestra con mayor detalle la evolución de la corriente consumida por el dispositivo durante un periodo de actividad (evento de conexión). En ella se distinguen una serie de fases o estados que se resumen a continuación:

1. Arranque: La CPU del dispositivo abandona el estado inactivo de bajo consumo.
2. Procesado (I): Se comienza a configurar la operación del transceptor radio.
3. Inicio de recepción: Se arranca el transceptor.
4. Recepción: El transceptor radio permanece en modo recepción para recibir un paquete desde el maestro.
5. Conmutación: El transceptor radio cambia el modo de operación de recepción a transmisión.
6. Transmisión: Si se recibe correctamente el paquete del maestro, se envía un paquete en respuesta.
7. Procesado(II): Se trata el paquete recibido y se realizan diversas acciones relacionadas con la pila de protocolo (como planificar el siguiente evento, etc).
8. Suspensión: La CPU se detiene y el dispositivo entra en bajo consumo.

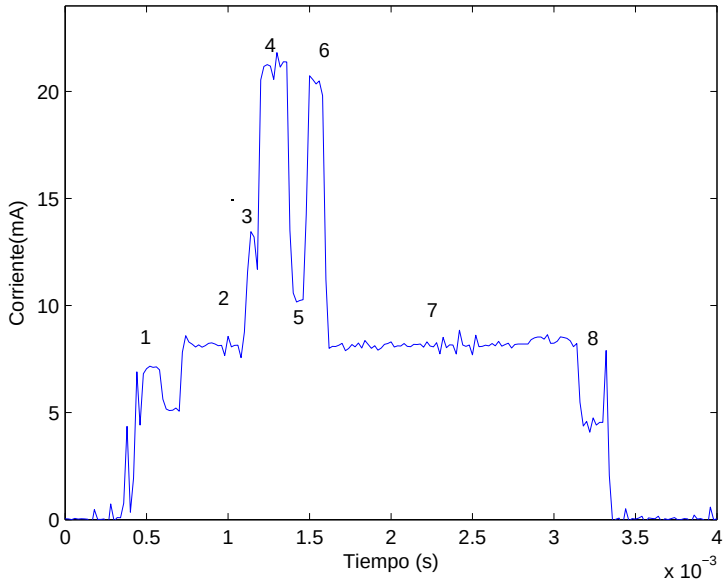


Figura 4.9: Detalle de la forma de onda de la corriente consumida por el dispositivo CC2541 durante un evento de conexión

La duración y el consumo de corriente medio aproximado observados durante las diferentes etapas se resumen en la tabla 4.1. La duración de alguno de los estados no es fija y viene determinada por distintos factores. Por ejemplo, la duración de la transmisión depende del tamaño del paquete, que es de 14 octetos para un paquete sin datos y de 21 octetos más el tamaño de los datos para un paquete con información del nivel de aplicación (debido a la sobrecarga de 4 y 3 octetos introducida por los protocolos L2CAP y ATT respectivamente).

Estado	Corriente media	Duración
Suspendido	30 μA	-
Arranque (1)	6 mA	0,4 ms
Procesado (2 y 7)	8,2 mA	1,8 ms
Inicio recepción (2)	13,5 mA	$\approx$ 0,08 ms
Recepción (4)	21,2 mA	0,2 ms
Conmutación (5)	10,5 mA	0,12 ms
Transmisión (6)	20,7 mA	variable según tamaño ($\approx$0,12 ms sin datos)
Suspensión (8)	5,5 mA	0,16 ms

Tabla 4.1: Consumo de las diferentes operaciones para el nodo basado en el CC2541.

4.4.2.1. Consumo en función del intervalo de conexión sin envío de datos

El funcionamiento periódico de *Bluetooth low energy* permite derivar un modelo de consumo sencillo para el modo conectado basado en el ciclo de actividad promedio durante cada intervalo de conexión. Realizando un razonamiento similar al planteado en el capítulo 3 para la tecnología *ZigBee*, el consumo medio de la conexión en ausencia de envío de datos puede estimarse mediante la ecuación (4.2):

$$I_{drain} = \frac{t_{act} \cdot I_{active} + (T_{interval} - t_{act}) \cdot I_{sleep}}{T_{interval}} \tag{4.2}$$

Donde I_{sleep} es la corriente media consumida mientras el dispositivo permanece inactivo, I_{active} es la corriente media durante cada evento de conexión, t_{act} es la duración media de dicho evento y $T_{interval}$ es la duración del intervalo de conexión. A su vez, I_{active} y t_{act} vienen dadas por:

$$t_{act} = \sum_{i=0}^{N} t_{state(i)} \tag{4.3}$$

$$I_{active} = \frac{\sum_{i=0}^{N} t_{state(i)} \cdot I_{state(i)}}{t_{act}} \tag{4.4}$$

Donde $I_{state(i)}$ es la corriente media consumida durante el estado i y $t_{state(i)}$ su duración.

4.4.2.2. Consumo en función del intervalo de conexión con envío de datos

En las subsecciones anteriores se ha considerado el consumo del dispositivo debido al mantenimiento de la conexión, es decir, sin envío de datos. En esta sección se analiza el consumo del dispositivo cuando está configurado para enviar información.

En *Bluetooth low energy* es el protocolo de gestión de atributos (ATT) el que determina cómo se intercambia información de nivel de aplicación entre dos dispositivos. Cada dispositivo ofrece una serie de atributos que pueden ser escritos o leídos por otros dispositivos (por ejemplo, en un sensor de temperatura, ésta sería un atributo que puede ser leído, mientras que parámetros que puedan afectar a la medida de temperatura como la velocidad de muestreo, la escala o la unidad empleada, serían atributos de escritura que pueden modificarse para controlar el funcionamiento del dispositivo). El protocolo (ATT) establece dos mecanismos diferentes para que un dispositivo cliente pueda recibir datos de

otro dispositivo servidor, la operación de lectura, que sería iniciada por el cliente y respondida por el servidor, y la operación de notificación, en la que el dispositivo cliente se suscribe a uno o varios atributos y a partir de entonces el servidor le envía la información a la que se ha suscrito cuando corresponde (por ejemplo, de forma periódica cada vez que se realiza una medida si se trata de un sensor). Este último es el mecanismo más eficiente y por tanto el que se ha considerado para su estudio, implementando en el dispositivo sensor un perfil de aplicación que envía una medida (de 2 octetos de longitud) cada 100ms una vez que el dispositivo cliente realiza la suscripción.

Las figuras 4.10 y 4.11 muestran el comportamiento del consumo de corriente del dispositivo sensor con el envío de datos activado para un intervalo de conexión de 1 segundo. Cada vez que se genera un nuevo dato (10 veces por segundo) el dispositivo abandona el bajo consumo, aunque no se realiza la transmisión de los datos hasta el comienzo del siguiente evento de conexión. La figura 4.11 muestra con mayor detalle el consumo durante un evento de conexión, en el que se realiza la transmisión de los 10 paquetes de datos generados desde el anterior evento de conexión. La transmisión de cada paquete va precedida siempre de la recepción de un paquete desde el dispositivo maestro indicando que está dispuesto a recibir o continuar recibiendo datos.

Para diferentes configuraciones de la duración del intervalo de conexión se observa que la actividad de generación de datos (determinada por la temporización de la generación de datos por parte de la aplicación) está intercalada con la actividad de transmisión de los mismos (definida por la duración del intervalo de conexión). El número de paquetes transmitidos en cada intervalo de conexión depende del número de paquetes de datos generados por la aplicación durante el intervalo de conexión, realizándose la transmisión de un paquete en cada intervalo de conexión cuando se genera uno o ningún paquete de datos de aplicación durante el mismo, y una transmisión por paquete de datos de nivel de aplicación si se generan más, con un límite máximo de hasta 10 envíos por cada intervalo de conexión. Cada paquete con datos consta de 21 bytes de cabecera y 2 bytes de datos de nivel de aplicación, por lo que su transmisión (a 1 Mbps) tiene una duración aproximada de unos $180\mu s$, mientras que los paquetes sin datos tienen una duración de unos $120\mu s$. En cuanto a la recepción, la primera recepción dentro de un evento de conexión tiene una duración mayor que las sucesivas, debido al ensanchamiento introducido para compensar la incertidumbre de la temporización de los dispositivos maestro y esclavo.

Si se asume que el consumo puede calcularse como el debido a la superposición de las actividades de sensado con la realizada en los eventos de conexión, la corriente media consumida vendrá dada por (4.5)

$$I_{drain} = r_{sen} \cdot I_{sen} + r_{con} \cdot I_{con} + (1 - r_{con} - r_{sen}) \cdot I_{sleep} \tag{4.5}$$

$$r_{sen} = \frac{t_{sen}}{T_{sen}} \tag{4.6}$$

$$r_{con} = \frac{t_{con}}{T_{interval}} \tag{4.7}$$

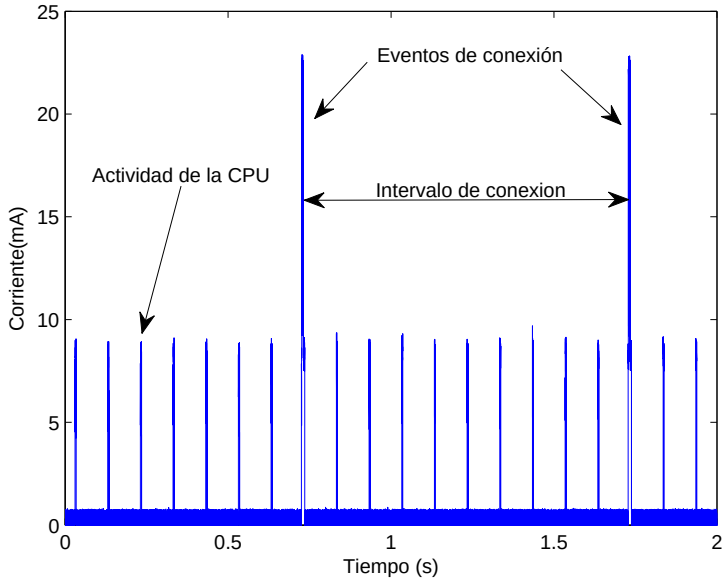


Figura 4.10: Forma de onda de la corriente consumida por el dispositivo CC2541 con el envío periódico activado

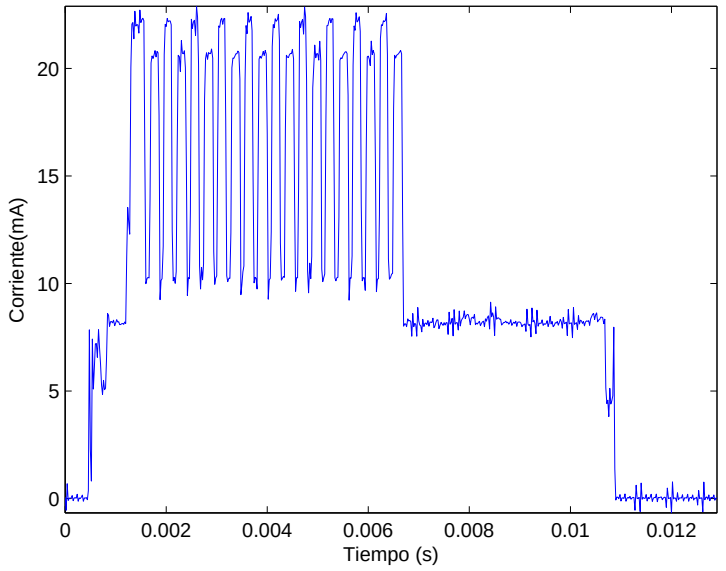


Figura 4.11: Detalle de la forma de onda de la corriente consumida durante un evento de conexión por el dispositivo CC2541 con el envío periódico activado

Donde I_{sen} sería la corriente media consumida durante cada evento de generación de datos, I_{con} la corriente media consumida en cada intervalo de conexión, t_{sen} la duración del periodo de actividad que tiene lugar cada vez que se generan datos, t_{con} la duración media del evento de conexión, T_{sen} la periodicidad con la que se generan los datos de nivel de aplicación y $T_{interval}$ la duración del intervalo de conexión. Los factores r_{con} y r_{sen} representarían la fracción del tiempo que el dispositivo permanece activo debido a la actividad en los eventos de conexión y en los eventos de generación de datos respectivamente.

El consumo medio para cada periodo de actividad puede calcularse mediante las ecuaciones (4.3) y (4.4) teniendo en cuenta los diferentes estados por los que pasa el dispositivo en cada tipo de evento. En el caso de los eventos de sensado, el dispositivo únicamente pasa por los estados de arranque, procesado (con una duración medida de 1.5ms aproximadamente) y suspensión. Para los eventos de conexión, pasa por todos los estados descritos en la sección anterior, siendo el número de intervalos de recepción, conmutación y transmisión proporcional a la relación entre $T_{interval}$ y T_{sen} (con un mínimo de una recepción, conmutación y transmisión para cada evento de conexión si $T_{sen} > T_{interval}$ y un máximo de 10).

4.4.3. Validación empírica de los modelos

Para validar el modelo propuesto, se ha realizado una batería de pruebas para medir el consumo promedio para diferentes valores del intervalo de conexión. Para realizar estas pruebas, el multímetro se configura para trabajar con la máxima resolución posible, utilizando intervalos de integración largos, y promediando posteriormente las medidas obtenidas durante un periodo de tiempo de operación. En todas las pruebas realizadas éste periodo ha sido configurado de forma que se garantice que su duración sea de al menos 15 minutos y contenga al menos 4000 eventos de conexión.

La figura 4.12 muestra el consumo estimado (utilizando el modelo) y el consumo medido (promediando con el multímetro) para el CC2541 en función del valor configurado para el intervalo de conexión (expresado en segundos), en ausencia de pérdidas y de intercambio de datos (el consumo sería debido por tanto únicamente a la actividad necesaria para mantener la conexión). El consumo estimado y medido en función del intervalo de conexión cuando el dispositivo genera datos de aplicación cada 100ms se muestra en la figura 4.13. En este último caso, parte del consumo se debe a la actividad periódica del nivel de aplicación y no a la comunicación *Bluetooth LE* en sí.

4.4.4. Efecto de las pérdidas

Los resultados mostrados en las secciones anteriores se han obtenido en ausencia de pérdidas y colisiones. La aparición de pérdidas de la comunicación puede aumentar el consumo de los dispositivos por diversas causas. En primer lugar la pérdida de eventos de conexión por parte del dispositivo esclavo aumenta la incertidumbre de la temporización, obligando a ensanchar la ventana de escucha al comienzo de cada intervalo de conexión. Por otra parte, si el parámetro *connSlaveLatency* está configurado a un valor superior a 0, cuando se produce una pérdida el dispositivo esclavo queda obligado a activarse en todos

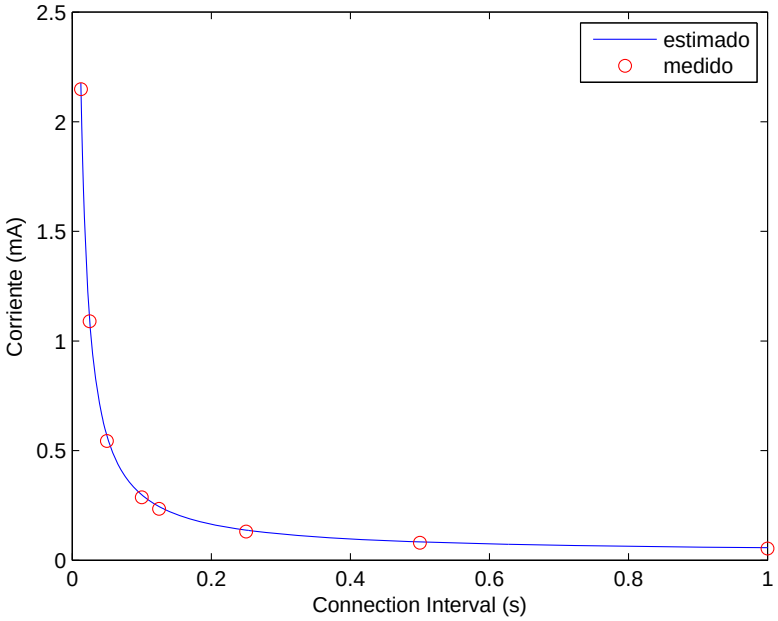


Figura 4.12: Consumo medio calculado y medido para el CC2541 en función del intervalo de conexión en ausencia de tráfico a nivel de aplicación

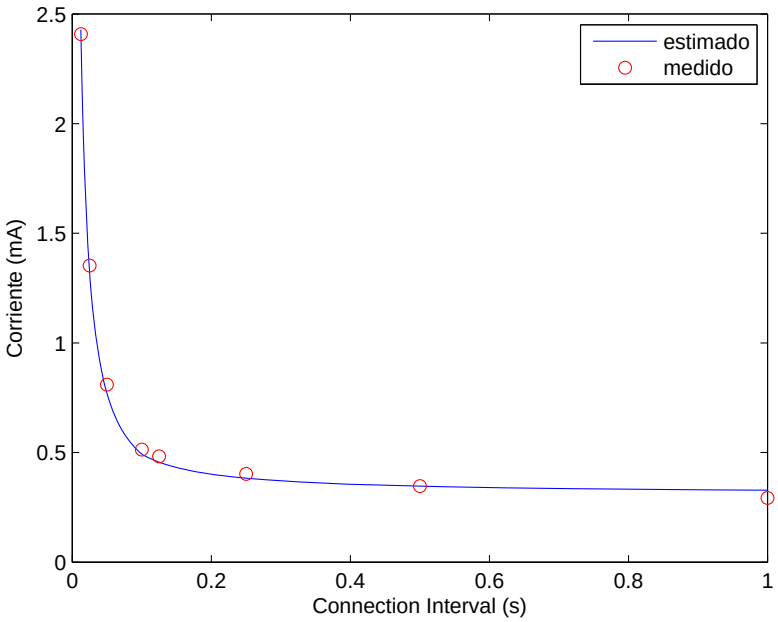


Figura 4.13: Consumo medio calculado y medido para el CC2541 en función del intervalo de conexión para tráfico generado por el nivel de aplicación cada 100ms

los eventos de conexión. Por último, la presencia de pérdidas repetidas podría provocar la finalización de la conexión, lo que obligaría a consumir energía en su restablecimiento.

Para completar el estudio se han realizado medidas del consumo medio introduciendo pérdidas de paquetes mediante el generador de ruido y el atenuador. La figura 4.14 muestra los resultados para diferentes valores del intervalo de conexión y del parámetro *latency* para una tasa de pérdidas de paquete (PER) aproximada de 0,35. Esta tasa de error de paquete (que incluye los paquetes de sondeo) es una estadística calculada y proporcionada por el propio dispositivo CC2541. En el eje de abcisas se representa el tiempo entre eventos de conexión en los que el dispositivo sensor debería idealmente despertarse (cada $latency + 1$ intervalos de conexión). En ausencia de pérdidas, el consumo para el mismo tiempo entre eventos de conexión en los que el dispositivo esclavo es similar para los diferentes valores de *latency*. Sin embargo, en presencia de pérdidas, el consumo aumenta con el valor de *latency* (pues el valor del intervalo de conexión es menor). En la figura también se observa un ligero aumento del consumo para $latency = 0$ con respecto al caso sin pérdidas, y que se debe al ensanchamiento de la ventana de recepción.

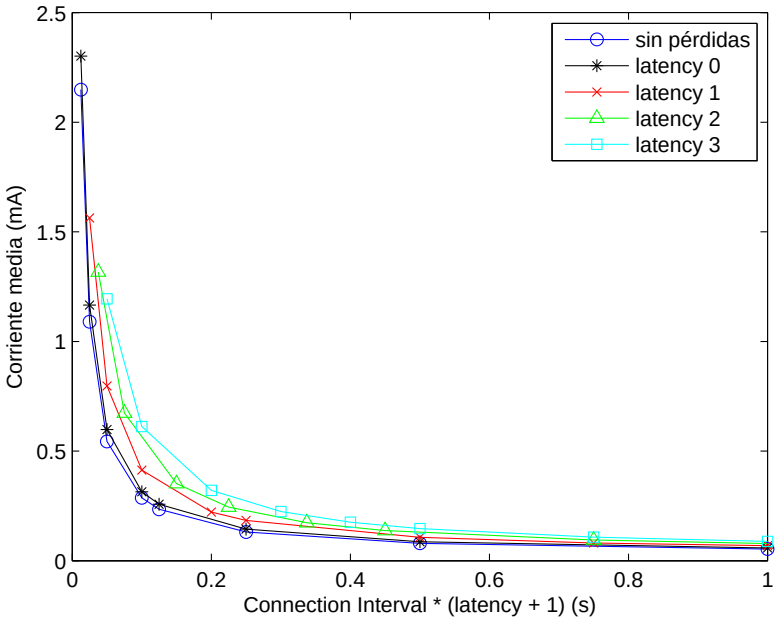


Figura 4.14: Consumo medio medido para el CC2541 en función del intervalo de conexión y del parámetro *latency* en presencia de pérdidas($PER \approx 0,35$) y sin tráfico a nivel de aplicación

4.5. Conclusiones

Bluetooth low energy es actualmente uno de los estándares con mayor proyección de futuro en el ámbito de las comunicaciones de área personal y se está posicionando como una de las piezas clave en el desarrollo de la IoT gracias a su amplio despliegue en dispositivos terminales y a la progresiva estandarización de optimizaciones y perfiles específicos para

ello.

En este capítulo se ha realizado una caracterización del consumo de un dispositivo *Bluetooth LE* y se ha propuesto un modelo para estimar el consumo de un dispositivo sensor considerando no sólo el consumo debido al mantenimiento de la conexión sino también el debido a la generación y envío de datos de nivel de aplicación. El modelo propuesto se ha validado mediante medidas empíricas en una plataforma de pruebas real.

Finalmente se han realizado algunas medidas adicionales sobre la plataforma real para comprobar el efecto de posibles pérdidas debidas a interferencias.

Capítulo 5

Conclusiones y Líneas Futuras

5.1. Conclusiones

La tesis se ha centrado principalmente en realizar una evaluación experimental de tres tecnologías para redes de área personal: *Bluetooth*, *ZigBee/802.15.4* y *Bluetooth Low Energy*. *Bluetooth* es una tecnología madura que está presente en la inmensa mayoría de terminales móviles y dispositivos de comunicación personales y permite el establecimiento de redes expontáneas con dispositivos vecinos. La tecnología 802.15.4/ZigBee es desde hace algunos años uno de los estándares más prometedores en el ámbito de la domótica y de las redes de sensores inalámbricas. *Bluetooth Low Energy* por su parte se plantea como un estándar emergente para permitir interactuar a los dispositivos de comunicación personales con objetos inteligentes alimentados a batería.

A continuación se resumen las principales aportaciones que se han obtenido a lo largo de los estudios presentados en los capítulos anteriores.

- Aportaciones en la evaluación de la tecnología *Bluetooth*

  - Tras realizar un exhaustivo análisis de los principios de operación de la tecnología, se ha realizado en primer lugar la caracterización del comportamiento energético de diversas implementaciones de dispositivos comerciales en diferentes modos de funcionamiento para las topologías *piconet* y *scatternet*.
  - De la caracterización realizada se ha derivado un modelo del comportamiento energético de los dispositivos cuando participan en más de una *piconet* para extender el área de cobertura de la red.
  - Se han realizado una serie de experimentos sobre dispositivos reales para estudiar su funcionamiento y prestaciones cuando se configuran para participar como puentes en una *scatternet*, ya que la norma apenas establece pautas sobre cómo deben comportarse los dispositivos en esta situación, comprobando que en la práctica su funcionamiento presenta bastante más limitaciones y peores prestaciones que las postuladas por algunos trabajos de investigación basados en análisis teóricos.
  - Finalmente se ha comprobado cómo afectan los modos de bajo consumo al

comportamiento de los dispositivos que actúan como puente en las diferentes configuraciones básicas.

- Aportaciones en la evaluación de la tecnología *ZigBee/802.15.4*

  - Tras un análisis detallado del funcionamiento de la tecnología, y en particular del control de acceso al medio, se ha realizado la caracterización del comportamiento energético de diversas implementaciones de dispositivos comerciales, considerando tanto las operaciones de transmisión de datos como algunas operaciones para el establecimiento y mantenimiento de la red.
  - Una vez realizada esta caracterización empírica del consumo, y considerando el comportamiento que tiene un dispositivo *end-device*, se han planteado una serie de expresiones matemáticas que permitirían estimar la duración de la batería de un dispositivo sensor en función del tamaño de los datos que envía y la periodicidad con la que lo hace. El modelo propuesto define ecuaciones para obtener la vida máxima, mínima y media esperada para la carga de la batería, teniendo en consideración las esperas introducidas por el algoritmo CSMA/CA aplicado por la capa MAC de 802.15.4.
  - El modelo propuesto ha sido validado mediante la realización de medidas en el sistema de pruebas real para diversas configuraciones, y se han presentado algunas medidas adicionales que extienden los casos previstos por el modelo. Los resultados muestran que aunque en condiciones ideales el tiempo de operación a batería de un sensor *end-device* con un bajo ciclo de trabajo puede ser teóricamente largo, escenarios con una gran densidad de nodos y/o la presencia de fuentes de interferencias de otras tecnologías puede provocar una degradación apreciable en el mismo.
  - Finalmente, se ha desarrollado una metodología para permitir la medida del retardo de transmisión entre nodos de una red *ZigBee*, que se ha utilizado para estudiar el impacto del ciclo de trabajo de los dispositivos *end-device* en las prestaciones de la red y que podría utilizarse para el estudio del efecto de otros parámetros de configuración.

- Aportaciones en la evaluación de la tecnología *Bluetooth LE*

  - Tras analizar con detalle las nuevas características y procedimientos introducidos para mejorar el consumo de batería con respecto a la tecnología *Bluetooth*, se ha caracterizado el consumo de algunos dispositivos comerciales al realizar distintas operaciones y con distintos parámetros de funcionamiento.
  - Se ha desarrollado un sistema de pruebas automático para evaluar el consumo de los dispositivos en presencia de pérdidas y diferentes tasas de tráfico, con objeto de establecer posibles límites en la duración de la batería en función de estos parámetros.

- Se ha propuesto, y validado utilizando el sistema de pruebas desarrollado, un modelo sencillo de consumo para dispositivos esclavos que considera tanto el gasto de batería debido al mantenimiento de la conexión como el debido a la actividad de envío de datos.

5.2. Líneas futuras

En este punto se proponen dos vertientes de trabajo. En un primer lugar, la vertiente derivada del desarrollo de los objetivos planteados en la tesis sobre la evaluación de las tecnologías bajo estudio, y en un segundo lugar, la derivada de la propia evolución de las tecnologías en el marco del paradigma de la Internet de las cosas.

Respecto a la primera vertiente, el desarrollo de los objetivos de la tesis para las tecnologías bajo estudio descarta totalmente la tecnología *Bluetooth* tradicional como una opción con futuro en el soporte de aplicaciones para la Internet de las cosas en las que tanto el bajo consumo como el número de nodos soportados por la red de área personal sean factores relevantes. En cambio, en el caso de las tecnologías *ZigBee/802.15.4* y *Bluetooth low energy* la caracterización de su consumo las hace valederas en primera instancia. No obstante, tal y como se ha dejado entrever a lo largo de la tesis existen puntos que es necesario continuar evaluando y que desencadenan las siguientes líneas de trabajo futuras:

- Estudio del retardo en redes *ZigBee*. Se ha desarrollado una metodología para permitir la realización de medidas de retardo sobre sistemas de pruebas reales y se ha realizado el estudio del retardo entre dos únicos nodos: el nodo *router* y el nodo *end-device*. Se propone, utilizando la metodología desarrollada en esta Tesis, extender el estudio del retardo a una red *ZigBee* con mayor número de nodos y múltiples saltos.

- Estudio de redes *Bluetooth low energy*. Aunque se ha realizado un estudio de cómo el consumo de esta tecnología puede verse afectado por la presencia de pérdidas, la plataforma utilizada (basada en un atenuador y generador de ruido AWGN) no permite un control muy preciso de la probabilidad de pérdida de paquetes, por lo que una posible línea para profundizar en este estudio sería seguir una estrategia similar a la implementada en el estudio de *ZigBee*. Además, el estudio del consumo que se ha desarrollado en esta tesis se ha llevado a cabo punto a punto, ya que el estándar no especifica otras topologías de red. Por ello, una futura línea de trabajo es aquella que permita el desarrollo de otras topologías y la evaluación de sus prestaciones.

- Estudio comparativo de redes *ZigBee* y *Bluetooth low energy*. En esta tesis se ha comprobado de forma experimental y teórica que ambas tecnologías están en el mismo orden de magnitud de consumo, por lo tanto, tras el desarrollo de la línea anterior, la comparación en términos de consumo y retardo a nivel de red es una posible línea de trabajo futuro. En este sentido, recientemente están comenzando a aparecer en el mercado dispositivos comerciales multiestándar como el CC2650 de Texas Instruments [Cc2d] que según el *firmware* con el que se programen pueden comportarse como dispositivos *ZigBee*, *6LowPAN* o *Bluetooth low energy*, lo que facilitaría la realización de este estudio comparativo.

Respecto a la segunda vertiente, tal y como se ha comentado en el capítulo 1, la tendencia actual parece converger hacia una creciente unificación de las comunicaciones entre dispositivos mediante la tecnología IPv6, en un marco de creciente proliferación de muy diversos tipos de dispositivos heterogéneos, con muy distintas capacidades de cómputo, memoria y batería, y con distintos requerimientos de calidad de servicio y prestaciones. 6LowPAN y otras propuestas del IETF, que en principio estaban especificados para funcionar sobre 802.15.4 están empezando a ser adoptadas por otras tecnologías (entre ellas *Bluetooth low energy*) y a ser implementadas por las diferentes plataformas desarrolladas por la industria.

Por otra parte en el mercado están empezando a aparecer dispositivos vestibles con bastante capacidad de proceso y con capacidad de conexión en red. La optimización del consumo de estos dispositivos para aumentar la duración de la operación de la batería va a ser uno de los factores determinantes en su adopción por parte del público general. A medida que su tecnología vaya mejorando y se reduzca el consumo debido a la operación del procesador, el consumo debido a las comunicaciones puede ir ganando en importancia.

Las redes inalámbricas de área personal y sus estándares están en continua evolución y constantemente están apareciendo nuevas mejoras y aplicaciones [VTPVG+14]. Una de las tecnologías más prometedoras [MVC+15] parece ser la modificación 802.15.4e recogida en la versión de 2015 del estándar (aunque estaba propuesta con anterioridad como añadido a la norma de 2011) y que introduce un esquema TDMA para permitir una operación más coordinada entre los dispositivos. A pesar de su relativa juventud, ya existen algunas implementaciones preliminares [Ope] y estudios sobre el funcionamiento y prestaciones de dicha tecnología en diferentes escenarios [VWC+14; SVW+14; SXZ15; ANK15]. Por otra parte ya se ha comentado que la evolución de *Bluetooth low energy* empieza a apuntar a la inclusión de un cierto soporte para redes malladas [RMHV15], por lo que poco a poco ambas ambos estándares se irán convirtiendo en más directos competidores.

5.3. Publicaciones

5.3.1. Artículos de revista relacionados con la tesis

[CCG+07] J.-C. Cano, J.-M. Cano, E. González, C. Calafate y P. Manzoni, "How does energy consumption impact performance in Bluetooth?", *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, n.º 3, pág. 7, 2007, ISSN: 01635999. DOI: 10.1145/1328690.1328694.

[CCGC+09] J. C. Cano, J. M. Cano-García, C. T. Calafate, E. González y P. Manzoni, "Trade-off between power consumption and performance in Bluetooth", *Ad-Hoc and Sensor Wireless Networks*, vol. 8, n.º 1-2, págs. 141-159, 2009, ISSN: 15519899. DOI: 10.1109/SENSORCOMM.2007.4394940.

[CCGCG10b] E. Casilari, J. M. Cano-García y G. Campos-Garrido, "Modeling of current consumption in 802.15.4/ZigBee sensor motes", *Sen-*

sors, vol. 10, n.º 6, págs. 5443-5468, 2010, ISSN: 1424-8220. DOI: 10.3390/s100605443.

[CCM+08] J.-C. Cano, C. Calafate, P. Manzoni, J.-M. Cano y E. Gonzalez, "Design and validation of a low-power network node for pervasive applications", *International Journal of Software Engineering and Its Applications (IJSEIA)*, vol. 2, n.º 1, págs. 21-32, oct. de 2008.

5.3.2. Comunicaciones a congresos relacionadas con la tesis

[CCA12] J. M. Cano-García, E. Casilari y F. Adbib, "A study on the effect of packet collisions on battery lifetime of 802.15.4 motes", en *Sensorcomm 2012*, 2012, págs. 209-214, ISBN: 9781612082073.

[CCG+06b] J. C. Cano, J. M. Cano, E. González, C. Calafate y P. Manzoni, "Power characterization of a Bluetooth-based wireless node for ubiquitous computing", *Second International Conference on Wireless and Mobile Communications, ICWMC 2006*, vol. 00, n.º c, 2006. DOI: 10.1109/ICWMC.2006.75.

[CCG10] E. Casilari y J. M. Cano-García, "Impact of the parameterization of IEEE 802.15.4 medium access layer on the consumption of ZigBee sensor motes", en *Ubicomm 2010*, 2010, págs. 117-123, ISBN: 9781612080000.

[CCG11] E Casilari y J. M. Cano-García, "An empirical evaluation of the consumption of 802.15.4/ZigBee sensor motes in noisy environments", en *IEEE International Conference on Networking, Sensing and Control*, 2011, págs. 11-13, ISBN: 9781424495733.

[CCGCG10a] E. Casilari, G. Campos-Garrido y J. M. Cano-García, "Characterization of battery consumption in 802.15.4/ZigBee sensor motes", en *IEEE International Symposium on Industrial Electronics*, 2010, págs. 3471-3476.

[CFLCG08] E Casilari, a. Flórez-Lara y J. Cano-García, "Analysis of the scalability of hierarchical IEEE 802.15.4/Zigbee networks", *Proceedings of the 3rd international conference on Scalable information systems*, 2008.

[CGCGP13] J. Cano-Garcia, E. Casilari y E. Gonzalez-Parada, "An empirical study on the performance of Bluetooth scatternets", en *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, jul. de 2013, págs. 1017-1022, ISBN: 978-1-4673-2480-9. DOI: 10.1109/IWCMC.2013.6583696.

[CHC09] E Casilari, J Hurtado Duenas y J. Cano García, "A study of policies for beacon scheduling in 802.15.4 cluster-tree networks", en *WSEAS International Conference on Applied Computer Science*, 2009, págs. 124-129, ISBN: 9789604741274.

5.3.3. Otras publicaciones en revista no relacionadas con la tesis

[CGGPC06] J. M. Cano-Garcia, E. Gonzalez-Parada y E. Casilari, "Experimental Analysis and Characterization of Packet Delay in UMTS Networks", en, Y. Koucheryavy, J. Harju y V. B. Iversen, eds., ép. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, mayo de 2006, vol. 4003, págs. 396-407, ISBN: 978-3-540-34429-2. DOI: 10.1007/11759355.

[GPCGDE05a] E. González-Parada, J. Cano-García y A. Díaz-Estrella, "A new methodology for TCP evaluation in a multiuser web environment", *Computer Communications*, vol. 28, n.º 4, págs. 449-465, mar. de 2005, ISSN: 01403664. DOI: 10.1016/j.comcom.2004.08.019.

[GPCGDE05b] E. González-Parada, J. M. Cano-García y A. Díaz-Estrella, "A new methodology for representation of TCP performance in multiconnection environments", *ACM SIGCOMM Computer Communication Review*, vol. 35, n.º 1, págs. 99-110, ene. de 2005, ISSN: 01464833. DOI: 10.1145/1052812.1052815.

[SPGPCG12] M. Santos-Pérez, E. González-Parada y J. M. Cano-García, "Topic-dependent language model switching for embedded automatic speech recognition", en, en *Ambient Intelligence-Software and Applications*, ép. Advances in Intelligent and Soft Computing, P. Novais, K. Hallenborg, D. I. Tapia y J. M. C. Rodríguez, eds., vol. 153, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, págs. 235-242, ISBN: 978-3-642-28782-4. DOI: 10.1007/978-3-642-28783-1.

[SPGPCg13] M. Santos-Perez, E. Gonzalez-Parada y J. Cano-garcia, "Mobile embodied conversational agent for task specific applications", English, *IEEE Transactions on Consumer Electronics*, vol. 59, n.º 3, 61o-614, ago. de 2013, ISSN: 0098-3063. DOI: 10.1109/TCE.2013.6626246.

Bibliografía

[ACD11] G. Anastasi, M. Conti y M. Di Francesco, "A comprehensive analysis of the MAC unreliability problem in IEEE 802.15.4 wireless sensor networks", English, *IEEE Transactions on Industrial Informatics*, vol. 7, n.º 1, págs. 52-65, feb. de 2011, ISSN: 1551-3203. DOI: 10.1109/TII.2010.2085440.

[ACDN10] G. Anastasi, M. Conti, M. Di Francesco y V. Neri, "Reliability and energy efficiency in multi-hop IEEE 802.15.4/ZigBee wireless sensor networks", en *The IEEE symposium on Computers and Communications*, IEEE, jun. de 2010, págs. 336-341, ISBN: 978-1-4244-7754-8. DOI: 10.1109/ISCC.2010.5546804.

[ADD09] M. Aiello, R. De Jong y J. De Nes, "Bluetooth broadcasting: how far can we go? an experimental study", en *2009 Joint Conferences on Pervasive Computing, JCPC 2009*, 2009, págs. 471-476, ISBN: 9781424452279. DOI: 10.1109/JCPC.2009.5420140.

[AFGM+15] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari y M. Ayyash, "Internet of things: a survey on enabling technologies, protocols and applications", *IEEE Communications Surveys & Tutorials*, vol. PP, n.º 99, págs. 1-1, 2015, ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2444095.

[ANK15] Y. Al-Nidawi y A. H. Kemp, "Mobility aware framework for timeslotted channel hopping IEEE 802.15.4e sensor networks", *IEEE Sensors Journal*, vol. 15, n.º 12, págs. 7112-7125, dic. de 2015, ISSN: 1530-437X. DOI: 10.1109/JSEN.2015.2472276.

[AY05] K. Akkaya y M. Younis, "A survey on routing protocols for wireless sensor networks", 2005. DOI: 10.1016/j.adhoc.2003.09.010. arXiv: 0112167 [hep-th].

[AY15] Z. Abbas y W. Yoon, "A survey on energy conserving mechanisms for the internet of things: wireless networking aspects", en, *Sensors*, vol. 15, n.º 10, págs. 24 818-24 847, sep. de 2015, ISSN: 1424-8220. DOI: 10.3390/s151024818.

[Ami11] M. Amiri, "Evaluation of power consumption and lifetime bounds", en *Wireless Sensor Networks*, LAP Lambert Academic Publishing, nov. de 2011, ISBN: 3846559776, 9783846559772.

[Arm] "mBed IoT device platform", ARM, inc., nov. de 2015.

[BBC13] J. D. Brock, R. F. Bruce y M. E. Cameron, "Changing the world with a raspberry pi", *Journal of Computing Sciences in Colleges*, vol. 29, n.º 2, págs. 151-153, 2013, ISSN: 1937-4771.

[BCD+08] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan y W. Dehaene, "Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives", en *Design, Automation, and Test in Europe: The Most Influential Papers of 10 Years Date*, 2008, págs. 221-234, ISBN: 0-7695-2288-2. DOI: 10.1109/DATE.2005.136.

[BCS+12] C. Bormann, A. P. Castellani, Z. Shelby, U. Bremen y Z. S. Sensinode, "CoAP: an application protocol for billions of tiny internet nodes", *IEEE Internet Computing*, vol. 16, n.º 2, págs. 62-67, 2012, ISSN: 10897801. DOI: 10.1109/MIC.2012.29.

[BDMT05] J. Beutel, M. Dyer, L. Meier y L. Thiele, "Scalable topology control for deployment-support networks", en *2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*, vol. 2005, IEEE, 2005, págs. 359-363, ISBN: 0-7803-9201-9. DOI: 10.1109/IPSN.2005.1440949.

[BDWL10] A. Bachir, M. Dohler, T. Watteyne y K. K. Leung, "Mac essentials for wireless sensor networks", *IEEE Communications Surveys and Tutorials*, vol. 12, n.º 2, págs. 222-248, 2010.

[BECL08] M. Blom, M. Ekstrom, J. Castano y M. Linden, "Bluetooth energy characteristics in wireless sensor networks", *2008 3rd International Symposium on Wireless Pervasive Computing*, págs. 198-202, 2008. DOI: 10.1109/ISWPC.2008.4556196.

[BFCE15] W. Bronzi, R. Frank, G. Castignani y T. Engel, "Bluetooth low energy performance and robustness analysis for inter-vehicular communications", *Ad Hoc Networks*, vol. 37, págs. 76-86, ago. de 2015, ISSN: 15708705. DOI: 10.1016/j.adhoc.2015.08.007.

[BGDP14] A. Betzler, C. Gomez, I. Demirkol y J. Paradells, "A holistic approach to ZigBee performance enhancement for home automation networks", *Sensors*, vol. 14, n.º 8, págs. 14 932-14 970, 2014, ISSN: 1424-8220. DOI: 10.3390/s140814932.

[BHG13] E. Baccelli, O. Hahm y M Günes, "RIOT OS: towards an OS for the internet of things", *Proc. of the 32nd IEEE . . .*, págs. 2453-2454, 2013. DOI: 10.1109/INFCOMW.2013.6970748.

[BK08] A. Bhatia y P. Kaushik, "A cluster based minimum battery cost AODV routing using multipath route for ZigBee", English, en *2008 16th IEEE International Conference on Networks*, IEEE, 2008, págs. 1-7, ISBN: 978-1-4244-3805-1. DOI: 10.1109/ICON.2008.4772594.

[BV09] C. Buratti y R. A. Verdone, "A mathematical model for performance of IEEE 802.15.4 beacon-enabled mode", en *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing Connecting the World Wirelessly - IWCMC '09*, 2009, pág. 1184, ISBN: 9781605585697. DOI: 10.1145/1582379.1582639.

[Beu05a] J Beutel, "Design and deployment of wireless networked embedded systems", Tesis doct., ETH Zurich, 2005.

[Beu05b] J. Beutel, "Robust topology formation using BTnodes", *Computer Communications*, vol. 28, n.º 13, págs. 1523-1530, ago. de 2005, ISSN: 01403664. DOI: 10.1016/j.comcom.2004.12.040.

[Ble] "Bluetooth low energy software stack", Texas Instruments, jun. de 2015.

[Blu92] L. Blumberg, "Multislope continuously integrating analog to digital converter", US Patent 5,148,171, sep. de 1992.

[Bou11] A Boulis, "Castalia: a simulator for wireless sensor networks and body area networks", inf. téc., 2011.

[CAA05] C. D. M. Cordeiro, S. Abhyankar y D. P. Agrawal, "An enhanced and energy efficient communication architecture for Bluetooth wireless PANs", *Ad Hoc Networks*, vol. 3, n.º 2, págs. 119-140, 2005, ISSN: 15708705. DOI: 10.1016/j.adhoc.2004.07.002.

[CB10] D. Contreras-Barcena, "Análisis y optimización del rendimiento en piconets Bluetooth para tráfico con requisitos de calidad", Tesis doct., Universidad Pontificia Comillas, 2010.

[CC06] C.-y. Chang y H.-r. Chang, "Adaptive role switching protocol for improving scatternet performance in Bluetooth radio networks", *IEEE Transactions on Consumer Electronics*, vol. 52, n.º 4, págs. 1229-1238, nov. de 2006, ISSN: 0098-3063. DOI: 10.1109/TCE.2006.273138.

[CC11a] D. Contreras y M. Castro, "Adaptive polling enhances quality and energy saving for multimedia over Bluetooth", *IEEE Communications Letters*, vol. 15, n.º 5, págs. 521-523, mayo de 2011, ISSN: 10897798. DOI: 10.1109/LCOMM.2011.031411.102476.

[CC11b] ——, "Impact of polling on Bluetooth piconet performance", *IEEE Communications Magazine*, vol. 49, n.º 9, págs. 84-89, sep. de 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.6011738.

[CC15] ——, "Experimental assessment of the adequacy of Bluetooth for opportunistic networks", *Ad Hoc Networks*, vol. 25, págs. 444-453, feb. de 2015, ISSN: 15708705. DOI: 10.1016/j.adhoc.2014.08.007.

[CCA12] J. M. Cano-García, E. Casilari y F. Adbib, "A study on the effect of packet collisions on battery lifetime of 802.15.4 motes", en *Sensorcomm 2012*, 2012, págs. 209-214, ISBN: 9781612082073.

[CCG+06a] J.-C. Cano, J.-M. Cano, E. González, C. Calafate y P. Manzoni, "Evaluation of the energetic impact of bluetooth low-power modes for ubiquitous computing applications", *Proceedings of the 3rd {ACM} international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks, mioCongreso*, págs. 1-8, 2006. DOI: 10.1145/1163610.1163612.

[CCG+06b] ——, "Power characterization of a Bluetooth-based wireless node for ubiquitous computing", *Second International Conference on Wireless and Mobile Communications, ICWMC 2006*, vol. 00, n.º c, 2006. DOI: 10.1109/ICWMC.2006.75.

[CCG+07] ——, "How does energy consumption impact performance in Bluetooth?", *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, n.º 3, pág. 7, 2007, ISSN: 01635999. DOI: 10.1145/1328690.1328694.

[CCG10] E. Casilari y J. M. Cano-García, "Impact of the parameterization of IEEE 802.15.4 medium access layer on the consumption of ZigBee sensor motes", en *Ubicomm 2010*, 2010, págs. 117-123, ISBN: 9781612080000.

[CCG11] E Casilari y J. M. Cano-García, "An empirical evaluation of the consumption of 802.15.4/ZigBee sensor motes in noisy environments", en *IEEE International Conference on Networking, Sensing and Control*, 2011, págs. 11-13, ISBN: 9781424495733.

[CCGC+09] J. C. Cano, J. M. Cano-García, C. T. Calafate, E. González y P. Manzoni, "Trade-off between power consumption and performance in Bluetooth", *Ad-Hoc and Sensor Wireless Networks*, vol. 8, n.º 1-2, págs. 141-159, 2009, ISSN: 15519899. DOI: 10.1109/SENSORCOMM.2007.4394940.

[CCGCG10a] E. Casilari, G. Campos-Garrido y J. M. Cano-García, "Characterization of battery consumption in 802.15.4/ZigBee sensor motes", en *IEEE International Symposium on Industrial Electronics*, 2010, págs. 3471-3476.

[CCGCG10b] E. Casilari, J. M. Cano-García y G. Campos-Garrido, "Modeling of current consumption in 802.15.4/ZigBee sensor motes", *Sensors*, vol. 10, n.º 6, págs. 5443-5468, 2010, ISSN: 1424-8220. DOI: 10.3390/s100605443.

[CCM+08] J.-C. Cano, C. Calafate, P. Manzoni, J.-M. Cano y E. Gonzalez, "Design and validation of a low-power network node for pervasive applications", *International Journal of Software Engineering and Its Applications (IJSEIA)*, vol. 2, n.º 1, págs. 21-32, oct. de 2008.

[CFLCG08] E Casilari, a. Flórez-Lara y J. Cano-García, "Analysis of the scalability of hierarchical IEEE 802.15.4/Zigbee networks", *Proceedings of the 3rd international conference on Scalable information systems*, 2008.

[CGCGP13] J. Cano-Garcia, E. Casilari y E. Gonzalez-Parada, "An empirical study on the performance of Bluetooth scatternets", en *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, jul. de 2013, págs. 1017-1022, ISBN: 978-1-4673-2480-9. DOI: 10.1109/IWCMC.2013.6583696.

[CGK01] A. Capone, M. Gerla y R. Kapoor, "Efficient polling schemes for Bluetooth picocells", en *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, vol. 7, IEEE, 2001, págs. 1990-1994, ISBN: 0-7803-7097-1. DOI: 10.1109/ICC.2001.936938.

[CHC09] E Casilari, J Hurtado Duenas y J. Cano García, "A study of policies for beacon scheduling in 802.15.4 cluster-tree networks", en *WSEAS International Conference on Applied Computer Science*, 2009, págs. 124-129, ISBN: 9789604741274.

[CKSG04] L.-J. Chen, R. Kapoor, M. Sanadidi y M. Gerla, "Enhancing bluetooth TCP throughput via link layer packet adaptation", English, en *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, vol. 7, IEEE, 2004, 4012-4016 Vol.7, ISBN: 0-7803-8533-0. DOI: 10.1109/ICC.2004.1313304.

[CLMM07] F. Cuomo, S. D. Luna, U. Monaco y T. Melodia, "Routing in ZigBee: benefits from exploiting the IEEE 802.15.4 association tree", *IEEE International Conference on Communications*, págs. 3271-3276, 2007, ISSN: 05361486. DOI: 10.1109/ICC.2007.542.

[CLWY07] Z. Chen, C. Lin, H. Wen y H. Yin, "An analytical model for evaluating IEEE 802.15.4 CSMA/CA protocol in low-rate wireless application model", en *Advanced Information Networking and Applications Workshops (AINAW'07)*, 2007, págs. 899-904, ISBN: 0769528473. DOI: 10.1109/AINAW.2007.77.

[CP15] M. Collotta y G. Pau, "A novel energy management approach for smart homes using Bluetooth low energy", *IEEE Journal on Selected Areas in Communications*, vol. 33, n.º 12, págs. 2988-2996, dic. de 2015, ISSN: 0733-8716. DOI: 10.1109/JSAC.2015.2481203.

[CPH+14] K. Cho, W. Park, M. Hong, G. Park, W. Cho, J. Seo y K. Han, "Analysis of latency performance of Bluetooth low energy (BLE) networks", en, *Sensors*, vol. 15, n.º 1, págs. 59-78, dic. de 2014, ISSN: 1424-8220. DOI: 10.3390/s150100059.

[CPMA14] R. C. Carrano, D. Passos, L. C. S. Magalhaes y C. V. N. Albuquerque, "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks", *IEEE Communications Surveys and Tutorials*, vol. 16, n.º 1, págs. 181-194, 2014, ISSN: 1553877X. DOI: 10.1109/SURV.2013.052213.00116.

[CSC06] L.-J. Chen, T. Sun e Y.-C. Chen, "Improving Bluetooth EDR data throughput using FEC and interleaving", en, en *International Conference on Mobile Ad-Hoc and Sensor Networks*, J. Cao, I. Stojmenovic, X. Jia y S. K. Das, eds., ép. Lecture Notes in Computer Science, vol. 4325, Berlin, Heidelberg: Springer Berlin Heidelberg, dic. de 2006, págs. 724-735, ISBN: 978-3-540-49932-9. DOI: 10.1007/11943952.

[CTK10] P. Casey, K. Tepe y N. Kar, "Design and implementation of a testbed for IEEE 802.15.4 (ZigBee) performance measurements", *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, n.º 1, pág. 103 406, abr. de 2010, ISSN: 1687-1499. DOI: 10.1155/2010/103406.

[CTS02] T. Chui, F. Thaler y W. Scanlon, "A novel channel modeling technique for performance analysis of Bluetooth baseband packets", *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, vol. 1, 2002, ISSN: 05361486. DOI: 10.1109/ICC.2002.996866.

[CWG+10] F. Chen, N. Wang, R. German, F. Dressler y C. Systems, "Simulation study of IEEE 802.15.4 LR-WPAN for industrial applications", *Wireless Communications and Mobile Computing*, vol. 621, n.º 5, págs. 609-621, 2010, ISSN: 15308669. DOI: 10.1002/wcm.

[Cc2a] "CC2520 second generation 2.4 ghz ZigBee/IEEE 802.15.4 RF transceiver", Texas Instruments, mayo de 2010.

[Cc2b] "CC2540 2.4-ghz bluetooth low energy system-on-chip", Texas Instruments, 2010.

[Cc2c] "CC2541: SimpleLink Bluetooth smart and proprietary wireless MCU", Texas Instruments, jun. de 2013.

[Cc2d] "CC2650 simplelink multi-standard 2.4 ghz ultra-low power wireless MCU", Texas Instruments, jun. de 2015.

[DAC+11] M. Di Francesco, G. Anastasi, M. Conti, S. K. Das y V. Neri, "Reliability and energy-efficiency in IEEE 802.15.4/ZigBee sensor networks: an adaptive and cross-layer approach", *IEEE Journal on Selected Areas in Communications*, vol. 29, n.º 8, págs. 1508-1524, sep. de 2011, ISSN: 0733-8716. DOI: 10.1109/JSAC.2011.110902.

[DGV04] A. Dunkels, B. Grönvall y T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors", *Proceedings - Conference on Local Computer Networks, LCN*, págs. 455-462, 2004, ISSN: 0742-1303. DOI: 10.1109/LCN.2004.38.

[DHM+07] D. Dubhashi, O. Häggström, G. Mambrini, A. Panconesi y C. Petrioli, "Bluepleiades, a new solution for device discovery and scatternet formation in multi-hop Bluetooth networks", *Wireless Networks*, vol. 13, n.º 1, págs. 107-125, 2007, ISSN: 10220038. DOI: 10.1007/s11276-006-1304-7.

[DHTS13] A. Dementyev, S. Hodges, S. Taylor y J. Smith, "Power consumption analysis of Bluetooth low energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario", en *2013 IEEE International Wireless Symposium (IWS)*, IEEE, abr. de 2013, págs. 1-4, ISBN: 978-1-4673-2141-9. DOI: 10.1109/IEEE-IWS.2013.6616827.

[DR13] A. Dahlstrom y R. Rajagopalan, "Performance analysis of routing protocols in ZigBee non-beacon enabled WSNs", *2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013*, págs. 932-937, 2013. DOI: 10.1109/CCNC.2013.6488587.

[DeC14] J. DeCuir, "Introducing Bluetooth smart: part i: a look at both classic and new technologies.", *IEEE Consumer Electronics Magazine*, vol. 3, n.º 1, págs. 12-18, 2014, ISSN: 2162-2248. DOI: 10.1109/MCE.2013.2284932.

[Dec14] J. Decuir, "Introducing Bluetooth smart: part ii: applications and updates.", *IEEE Consumer Electronics Magazine*, vol. 3, n.º 2, págs. 25-29, 2014, ISSN: 2162-2248. DOI: 10.1109/MCE.2013.2297617.

[Dec15] ——, "Bluetooth smart support for 6LoBTLE: applications and connection questions.", *IEEE Consumer Electronics Magazine*, vol. 4, n.º 2, págs. 67-70, abr. de 2015, ISSN: 2162-2248. DOI: 10.1109/MCE.2015.2392955.

[Dun03] A. Dunkels, "TCP/IP for 8-bit architectures", en *Proceedings of the 1st international conference on Mobile systems, applications and services - MobiSys '03*, 2003, págs. 85-98. DOI: 10.1145/1066116.1066118.

[EA15] J Etxaniz y G Aranguren, "Modeling of the data transportation network of a multi-hop data-content-sharing home network", *Consumer Electronics, IEEE Transactions on*, vol. 61, n.º 1, págs. 31-38, 2015, ISSN: 0098-3063. DOI: 10.1109/TCE.2015.7064108.

[EBL+12] M. C. Ekstrom, M. Bergblomma, M. Linden, M. Bjorkman y M. Ekstrom, "A Bluetooth radio energy consumption model for low-duty-cycle applications", *IEEE Transactions on Instrumentation and Measurement*, vol. 61, n.º 3, págs. 609-617, mar. de 2012, ISSN: 0018-9456. DOI: 10.1109/TIM.2011.2172997.

[EMA14] J Etxaniz, P. M. Monje y G Aranguren, "Note: methodology for the analysis of Bluetooth gateways in an implemented scatternet.", *The Review of scientific instruments*, vol. 85, n.º 3, pág. 036 111, mar. de 2014, ISSN: 1089-7623. DOI: 10.1063/1.4869016.

[FC02] C.-C. Foo y K.-C. Chua, "Bluerings - bluetooth scatternets with ring structures", en *Proceeding Wireless and Optical Communications*, 2002.

[FJYH12] Facheng Fang, Jun Sun, Yaliang Han y Hongbo Zhu, "Performance analysis of ZigBee under WLAN and multiple Bluetooth piconets interferences", en *IET International Conference on Information Science and Control Engineering 2012 (ICISCE 2012)*, Institution of Engineering y Technology, 2012, págs. 2.44-2.44, ISBN: 978-1-84919-641-3. DOI: 10.1049/cp.2012.2357.

[FKK13] R. Friedman, A. Kogan e Y. Krivolapov, "On power and throughput tradeoffs of WiFi and Bluetooth in smartphones", *IEEE Transactions on Mobile Computing*, vol. 12, n.º 7, págs. 1363-1376, jul. de 2013, ISSN: 1536-1233. DOI: 10.1109/TMC.2012.117.

[FLMA+09] H. Fernandez-Lopez, P. Macedo, J. Afonso, J. Correia y R. Simoes, "Performance evaluation of a ZigBee-based medical sensor network", English, en *2009 3rd International Conference on Pervasive Computing Technologies for Healthcare*, ICST, 2009, págs. 1-4, ISBN: 978-963-9799-42-4. DOI: 10.4108/ICST.PERVASIVEHEALTH2009.6002.

[FVV12] N. Fourty, A. Van Den Bossche y T. Val, "An advanced study of energy consumption in an IEEE 802.15.4 based network: everything but the truth on 802.15.4 node lifetime", *Computer Communications*, vol. 35, n.º 14, págs. 1759-1767, 2012, ISSN: 01403664. DOI: 10.1016/j.comcom.2012.05.008.

[Fuh02] P. Fuhrman, "Analog-to-digital converter", US Patent 6,384,760, mayo de 2002.

[Fur10] K Fursan, "Inside Bluetooth low energy technology", *Embedded*, nov. de 2010.

[GCR05] N. Golmie, D. Cypher y O. Rebala, "Performance analysis of low rate wireless technologies for medical applications", *Computer Communications*, vol. 28, n.º 10, págs. 1266-1275, jun. de 2005, ISSN: 01403664. DOI: 10.1016/j.comcom.2004.07.021.

[GDP11] C. Gomez, I. Demirkol y J. Paradells, "Modeling the maximum throughput of Bluetooth low energy in an error-prone link", *IEEE Communications Letters*, vol. 15, n.º 11, págs. 1187-1189, 2011, ISSN: 10897798. DOI: 10.1109/LCOMM.2011.092011.111314.

[GLB+03] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer y D. Culler, "The nesc language: a holistic approach to networked embedded systems", *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation - PLDI '03*, pág. 1, 2003, ISSN: 03621340. DOI: 10.1145/781131.781133.

[GOP12] C. Gomez, J. Oller y J. Paradells, "Overview and evaluation of Bluetooth low energy: an emerging low-power wireless technology", *Sensors (Switzerland)*, vol. 12, n.º 9, págs. 11 734-11 753, 2012, ISSN: 14248220. DOI: 10.3390/s120911734.

[GRX+11] M. Goyal, D. Rohm, W. Xie, S. H. Hosseini, K. S. Trivedi, Y. Bashir y a. Divjak, "A stochastic model for beaconless IEEE 802.15.4 MAC operation", *Computer Communications*, vol. 34, n.º 12, págs. 1460-1474, 2011, ISSN: 01403664. DOI: 10.1016/j.comcom.2010.12.004.

[GYYL09] K. Gill, S.-H. Yang, F. Yao y X. Lu, "A ZigBee-based home automation system", *IEEE Transactions on Consumer Electronics*, vol. 55, n.º 2, págs. 422-430, mayo de 2009, ISSN: 0098-3063. DOI: 10.1109/TCE.2009.5174403.

[Goe92] W. Goeke, "Continuously integrating high-resolution analog-to-digital converter", US Patent 5,117,227, mayo de 1992.

[Gol04] N. Golmie, "Bluetooth dynamic scheduling and interference mitigation", en, *Mobile Networks and Applications*, vol. 9, n.º 1, págs. 21-31, 2004, ISSN: 1383469X. DOI: 10.1023/A:1027313621955.

[HAMMG09] J. Hipolito, N. Arballo, J. Michel-Macarty y E. Garcia, "Bluetooth performance analysis in wireless personal area networks", en *2009 Electronics, Robotics and Automotive Mechanics Conference (CERMA)*, 2009, págs. 38-43, ISBN: 978-0-7695-3799-3. DOI: 10.1109/CERMA.2009.48.

[HHP15] J. Heuer, J. Hund y O. Pfaff, "Toward the web of things: applying web technologies to the physical world", *Computer*, vol. 48, n.º 5, págs. 34-42, mayo de 2015, ISSN: 0018-9162. DOI: 10.1109/MC.2015.152.

[HKC08] T. Hassan, A. Kayssi y A. Chehab, "Ring of masters (RoM): a new ring structure for Bluetooth scatternets with dynamic routing and adaptive scheduling schemes", *Pervasive Mobile Computers*, vol. 4, n.º 4, págs. 546-561, 2008.

[HKPZ12] Y.-K. Huang, C.-F. Kuo, A.-C. Pang y W. Zhuang, "Stochastic delay guarantees in ZigBee cluster-tree networks", English, en *2012 IEEE International Conference on Communications (ICC)*, IEEE, jun. de 2012, págs. 4926-4930, ISBN: 978-1-4577-2053-6. DOI: 10.1109/ICC.2012.6363961.

[HL10a] D.-M. Han y J.-H. Lim, "Design and implementation of smart home energy management systems based on ZigBee", *IEEE Transactions on Consumer Electronics*, vol. 56, n.º 3, págs. 1417-1425, ago. de 2010, ISSN: 0098-3063. DOI: 10.1109/TCE.2010.5606278.

[HL10b] C. F. Hsu y C. Y. Liu, "An adaptive traffic-aware polling and scheduling algorithm for Bluetooth piconets", *IEEE Transactions on Vehicular Technology*, vol. 59, n.º 3, págs. 1402-1414, 2010, ISSN: 00189545. DOI: 10.1109/TVT.2009.2038271.

[HPH+12] Y.-K. Huang, A.-C. Pang, P.-C. Hsiu, W. Zhuang y L. Pangfeng, "Distributed throughput optimization for ZigBee cluster-tree networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, n.º 3, págs. 513-520, mar. de 2012, ISSN: 1045-9219. DOI: 10.1109/TPDS.2011.192.

[ICT+13] I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Abeele, E. Poorter, I. Moerman y P. Demeester, "IETF standardization in the field of the internet of things (IoT): a survey", *Journal of Sensor and Actuator Networks*, vol. 2, n.º 2, págs. 235-287, 2013, ISSN: 2224-2708. DOI: 10.3390/jsan2020235.

[Ieea] "IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)", *IEEE Std 802.15.4-2003*, págs. 1-670, 2003. DOI: 10.1109/IEEESTD.2003.94389.

[Ieeb] "IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (WPANs)", *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, págs. 1-320, sep. de 2006. DOI: 10.1109/IEEESTD.2006.232110.

[Ieec] "IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (LR-WPANs)", *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, págs. 1-314, sep. de 2011. DOI: 10.1109/IEEESTD.2011.6012487.

[Ieed] "IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (LR-WPANs)", *IEEE P802.15.4-REVc/D00 (Revision of IEEE Std 802.15.4-2011)*, págs. 1-684, 2015.

[Int] "Intel curie module: unleashing wearable device innovation", Intel, inc., nov. de 2015.

[JCG06] S. Jung, A. Chang y M. Gerla, "Comparison of Bluetooth interconnection methods using blueprobe", *2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, WiOpt 2006*, 2006. DOI: 10.1109/WIOPT.2006.1666487.

[JCJM14] A Jedda, A Casteigts, G. Jourdan y H. Mouftah, "Bluetooth scatternet formation from a time-efficiency perspective", *Wireless networks*, 2014.

[JZLZ13] M. Javed, K. Zen, H. B. Lenando y H. Zen, "Performance evaluation of beacon enabled IEEE 802.15.4 MAC for mobile wireless sensor networks under ns-2", en *2013 8th International Conference on Information Technology in Asia (CITA)*, IEEE, jul. de 2013, págs. 1-7, ISBN: 978-1-4799-1092-2. DOI: 10.1109/CITA.2013.6637566.

[KA05] D. N. Kalofonos y S. Asthana, "A Bluetooth scatternet formation and healing protocol for ad-hoc group collaboratioa", English, en *2nd International Conference on Broadband Networks, BROADNETS 2005*, vol. 2005, IEEE, 2005, págs. 832-835, ISBN: 0-7803-9276-0. DOI: 10.1109/ICBN.2005.1589684.

[KBG05] C. Kallo, M. Brunato y M. Gerla, "Throughput, energy and path length tradeoffs in Bluetooth scatternets", en *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, vol. 5, IEEE, 2005, págs. 3319-3323, ISBN: 0-7803-8938-7. DOI: 10.1109/ICC.2005.1495036.

[KCJ+07] C. Kiss Kalló, C. F. Chiasserini, S. Jung, M. Brunato y M. Gerla, "Hop count based optimization of Bluetooth scatternets", *Ad Hoc Networks*, vol. 5, n.º 3, págs. 340-359, 2007, ISSN: 15708705. DOI: 10.1016/j.adhoc.2005.12.001.

[KGM14] M. Khanafer, M. Guennoun y H. T. Mouftah, "A survey of beacon-enabled IEEE 802.15.4 MAC protocols in wireless sensor networks", *IEEE Communications Surveys & Tutorials*, vol. 16, n.º 2, págs. 856-876, 2014, ISSN: 1553-877X. DOI: 10.1109/SURV.2013.112613.00094.

[KKHH06] M. Kohvakka, M. Kuorilehto, M. Hännikäinen y T. D. Hämäläinen, "Performance analysis of IEEE 802.15.4 and ZigBee for large-scale wireless sensor network applications", en *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, New York, New York, USA: ACM Press, oct. de 2006, págs. 48-57, ISBN: 1595934871. DOI: 10.1145/1163610.1163619.

[KL11] S. Kamath y J. Lindh, "Measuring Bluetooth low energy power consumption", Texas instruments application note AN092, 2011.

[KLJ15] H.-S. Kim, J. Lee y J. W. Jang, "BLEmesh: a wireless mesh network protocol for bluetooth low energy devices", en *2015 3rd International Conference on Future Internet of Things and Cloud*, IEEE, ago. de 2015, págs. 558-563, ISBN: 978-1-4673-8103-1. DOI: 10.1109/FiCloud.2015.21.

[Kir14] S. Kirk, "The wearables revolution: is standardization a help or a hindrance?: mainstream technology or just a passing phase?", *IEEE Consumer Electronics Magazine*, vol. 3, n.º 4, págs. 45-50, oct. de 2014, ISSN: 2162-2248. DOI: 10.1109/MCE.2014.2345996.

[LCM12a] J. Liu, C. Chen e Y. Ma, "Modeling neighbor discovery in Bluetooth low energy networks", *IEEE Communications Letters*, vol. 16, n.º 9, págs. 1439-1441, sep. de 2012, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2012.073112.120877.

[LCM12b] ——, "Modeling and performance analysis of device discovery in Bluetooth low energy networks", en *GLOBECOM - IEEE Global Telecommunications Conference*, 2012, págs. 1538-1543, ISBN: 9781467309219. DOI: 10.1109/GLOCOM.2012.6503332.

[LCMX13] J. Liu, C. Chen, Y. Ma e Y. Xu, "Energy analysis of device discovery for Bluetooth low energy", en *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, IEEE, sep. de 2013, págs. 1-5, ISBN: 978-1-4673-6187-3. DOI: 10.1109/VTCFall.2013.6692181.

[LCXW11] Y Li, H Chen, R Xie y J. Wang, "BGN: a novel scatternet formation algorithm for Bluetooth-based sensor networks", *Mobile Information Systems*, 2011.

[LG10] J. R. Luque-Giráldez, "Modelado del retardo de transmisión en bluetooth 2.0+EDR", Tesis doct., Universidad de Málaga, 2010.

[LL06] J. Li y X. Liu, "A collision resolution technique for robust coexistence of multiple Bluetooth piconets", en *IEEE Vehicular Technology Conference*, 2006, págs. 1087-1091, ISBN: 1424400635. DOI: 10.1109/VTCF.2006.230.

[LL09] S. H. Lee e Y. H. Lee, "Adaptive frequency hopping for Bluetooth robust to WLAN interference", *IEEE Communications Letters*, vol. 13, n.º 9, págs. 628-630, 2009, ISSN: 10897798. DOI: 10.1109/LCOMM.2009.090115.

[LLS03] Y. Liu, M. J. Lee y T. N. Saadawi, "A Bluetooth scatternet-route structure for multihop ad hoc networks", *IEEE Journal on Selected Areas in Communications*, vol. 21, n.º 2, págs. 229-239, 2003, ISSN: 07338716. DOI: 10.1109/JSAC.2002.807338.

[LLWC03] P. Levis, N. Lee, M. Welsh y D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications", *Proceedings of the 1st international conference on Embedded networked sensor systems*, págs. 126-137, 2003. DOI: 10.1145/958491.958506.

[LMC10] R. Luque, M. Moron y E. Casilari, "Minimum transmission delay in Bluetooth 2.0+EDR", English, *Electronics Letters*, vol. 46, n.º 13, pág. 955, jun. de 2010, ISSN: 00135194. DOI: 10.1049/el.2010.1108.

[LMC12] J. Luque, M. Morón y E. Casilari, "Analytical and empirical evaluation of the impact of gaussian noise on the modulations employed by Bluetooth enhanced data rates", en, *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, n.º 1, pág. 94, 2012, ISSN: 1687-1499. DOI: 10.1186/1687-1499-2012-94.

[LMC15] J.-R. Luque, M.-J. Morón y E. Casilari, "A characterization of the performance of Bluetooth 2.x+EDR technology in noisy environments", en, *Wireless Networks*, vol. 21, n.º 6, págs. 1969-1984, ene. de 2015, ISSN: 1022-0038. DOI: 10.1007/s11276-015-0888-1.

[LMP+05] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, a. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer y D. Culler, "TinyOS: an operating system for sensor networks", *Ambient Intelligence*, págs. 115-148, 2005. DOI: 10.1007/3-540-27139-2{_}7.

[LSS07] J.-S. Lee, Y.-W. Su y C.-C. Shen, "A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi", en *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2007, págs. 46-51, ISBN: 1-4244-0783-4. DOI: 10.1109/IECON.2007.4460126.

[LSW04] X. Y. Li, I. Stojmenovic e Y. Wang, "Partial delaunay triangulation and degree limited localized Bluetooth scatternet formation", *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, n.º 4, págs. 350-361, 2004, ISSN: 10459219. DOI: 10.1109/TPDS.2004.1271184.

[LTCT03] T. yu Lin, Y. chee Tseng, K. ming Chang y C. liang Tu, "Formation routing and maintenance protocols for the bluering scatternet of bluetooth", en *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003, págs. 313-322.

[LTT15] J.-r. Lin, T. Talty y O. Tonguz, "On the potential of Bluetooth low energy technology for vehicular applications", *IEEE Communications Magazine*, vol. 53, n.º 1, págs. 267-275, ene. de 2015, ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7010544.

[LWG05] O. Landsiedel, K. Wehrle y S. Gotz, "Accurate prediction of power consumption in sensor networks", en *The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II.*, IEEE, 2005, págs. 37-44, ISBN: 0-7803-9246-9. DOI: 10.1109/EMNETS.2005.1469097.

[LWS12] J.-s. Lee, Y.-M. Wang y C.-C. Shen, "Performance evaluation of ZigBee-based sensor networks using empirical measurements", en *IEEE International Conference on Cyber Technology in Automation*, IEEE, mayo de 2012, págs. 58-63, ISBN: 978-1-4673-1421-3. DOI: 10.1109/CYBER.2012.6392527.

[LYWA15] B.-H. Lee, E. Yundra, H.-K. Wu y M. U. H. Al Rasyid, "Analysis of superframe duration adjustment scheme for IEEE 802.15.4 networks", *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, n.º 1, pág. 103, abr. de 2015, ISSN: 1687-1499. DOI: 10.1186/s13638-015-0296-3.

[LZZ+10] M. J. Lee, R. Zhang, J. Zheng, G. S. Ahn, C. Zhu, T. R. Park, S. R. Cho, C. S. Shin y J. S. Ryu, "Ieee 802.15.5 wpan mesh standard-low rate part: meshing the wireless sensor networks", *IEEE Journal on Selected Areas in Communications*, vol. 28, n.º 7, págs. 973-983, 2010, ISSN: 07338716. DOI: 10.1109/JSAC.2010.100902.

[Lee06] J.-S. Lee, "Performance evaluation of IEEE 802.15.4 for low-rate wireless personal area networks", English, *IEEE Transactions on Consumer Electronics*, vol. 52, n.º 3, págs. 742-749, ago. de 2006, ISSN: 0098-3063. DOI: 10.1109/TCE.2006.1706465.

[MC04] T. Melodia y F. Cuomo, "Locally optimal scatternet topologies for bluetooth ad hoc networks", *Wireless On-demand Network Systems*, vol. 2928, págs. 116-129, 2004.

[MCM04] J. Misic, K. Chan y V. Misic, "Admission control in Bluetooth piconets", *IEEE Transactions on Vehicular Technology*, vol. 53, n.º 3, págs. 890-911, mayo de 2004, ISSN: 0018-9545. DOI: 10.1109/TVT.2004.827154.

[MF08] M. J. Morón-Fernández, "Estudio del rendimiento de perfiles bluetooth en redes de área personal", Tesis doct., Universidad de Málaga, 2008.

[MGS+05] T. K. Madsen, F. Gudmundsson, S. Sverrisson, H. P. Schwefel y R. Prasad, "Bluetooth scatternet with infrastructure support: Formation algorithms", *Consumer Communications and Networking Conference*, 2005.

[MH15] K. Mikhaylov y T. Hänninen, "Mechanisms for improving throughput and energy efficiency of Bluetooth low energy for multi node environment", *Journal of High Speed Networks*, vol. 21, n.º 3, págs. 165-180, ago. de 2015, ISSN: 18758940. DOI: 10.3233/JHS-150518.

[MHQ09] K. Morsi, X. Huagang y G. Qiang, "Performance estimation and evaluation of Bluetooth frequency hopping selection kernel", en *2009 Joint Conferences on Pervasive Computing, JCPC 2009*, IEEE, dic. de 2009, págs. 461-466, ISBN: 9781424452279. DOI: 10.1109/JCPC.2009.5420142.

[MKM03] V. B. Mišić, E. W. S. Ko y J. Mišić, "Adaptive cycle-limited scheduling scheme for Bluetooth piconets", en *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, vol. 2, 2003, págs. 1064-1068, ISBN: 0780378229. DOI: 10.1109/PIMRC.2003.1260274.

[MKSM11] P. Moravek, D. Komosny, M. Simek y L. Mraz, "Energy demands of 802.15.4/ZigBee communication with IRIS sensor motes", en *2011 34th International Conference on Telecommunications and Signal Processing (TSP)*, IEEE, ago. de 2011, págs. 69-73, ISBN: 978-1-4577-1410-8. DOI: 10.1109/TSP.2011.6043770.

[MLC10] M. J. Morón, R. Luque y E. Casilari, "Modeling of the transmission delay in Bluetooth piconets under serial port profile", *IEEE Transactions on Consumer Electronics*, vol. 56, n.º 4, págs. 2080-2085, 2010, ISSN: 00983063. DOI: 10.1109/TCE.2010.5681075.

[MLCDE07] M. Moron, R. Luque, E. Casilari y A. Diaz-Estrella, "An analytical study of the delay in Bluetooth networks using the personal area network profile", English, *IEEE Communications Letters*, vol. 11, n.º 11, págs. 845-847, nov. de 2007, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2007.071117.

[MLCDE08] M. Morón, R. Luque, E. Casilari y A. Díaz-Estrella, "Minimum delay bound in Bluetooth transmissions with serial port profile", English, *Electronics Letters*, vol. 44, n.º 18, pág. 1099, ago. de 2008, ISSN: 00135194. DOI: 10.1049/el:20081440.

[MLCDE09] M. J. Morón, R. Luque, E. Casilari y a. Díaz-Estrella, "Characterization of Bluetooth packet delay in noisy environments", *IEEE Communications Letters*, vol. 13, n.º 9, págs. 661-663, 2009, ISSN: 10897798. DOI: 10.1109/LCOMM.2009.091142.

[MM03a] J. Mišić y V. B. Mišić, "Bridges of Bluetooth county: topologies, scheduling, and performance", *IEEE Journal on Selected Areas in Communications*, vol. 21, n.º 2, págs. 240-258, 2003, ISSN: 07338716. DOI: 10.1109/JSAC.2002.807340.

[MM03b] ——, "Modeling Bluetooth piconet performance", *IEEE Communications Letters*, vol. 7, n.º 1, págs. 18-20, 2003, ISSN: 10897798. DOI: 10.1109/LCOMM.2002.807435.

[MM04a] J. Misic y V. Misic, "Bluetooth master/slave bridge scheduling with and without rendezvous points", en *24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings.*, IEEE, 2004, págs. 742-747, ISBN: 0-7695-2087-1. DOI: 10.1109/ICDCSW.2004.1284116.

[MM04b] V. B. Mišić y J. Mišić, "Performance of Bluetooth bridges in scatternets with limited service scheduling", *Mobile Networks and Applications*, vol. 9, n.º 1, págs. 73-87, 2004, ISSN: 1383469X. DOI: 10.1023/A:1027325907843.

[MM05] J. Misic y V. Misic, "TCP traffic in Bluetooth 1.2: performance and dimensioning of flow control", en *IEEE Wireless Communications and Networking Conference, 2005*, vol. 3, IEEE, 2005, págs. 1798-1804, ISBN: 0-7803-8966-2. DOI: 10.1109/WCNC.2005.1424785.

[MMK04] J. Misic, V. Misic y E. Ko, "Fixed cycles and adaptive bandwidth allocation can coexist in Bluetooth", *Canadian Journal of Electrical and Computer Engineering*, vol. 29, n.º 1, 2004, ISSN: 0840-8688. DOI: 10.1109/CJECE.2004.1425807.

[MMR05] J. Misic, V. Misic y G. Reddy, "On the performance of Bluetooth scatternets with finite buffers", en *25th IEEE International Conference on Distributed Computing Systems Workshops*, IEEE, 2005, págs. 865-870, ISBN: 0-7695-2328-5. DOI: 10.1109/ICDCSW.2005.105.

[MN06] D. Macii y L. Negri, "An automatic power consumption measurement procedure for Bluetooth modules", en *2006 IEEE Instrumentation and Measurement Technology Conference Proceedings*, IEEE, abr. de 2006, págs. 1182-1187, ISBN: 0-7803-9360-0. DOI: 10.1109/IMTC.2006.328446.

[MP07] D. Macii y D. Petri, "An effective power consumption measurement procedure for Bluetooth wireless modules", English, *IEEE Transactions on Instrumentation and Measurement*, vol. 56, n.º 4, págs. 1355-1364, ago. de 2007, ISSN: 0018-9456. DOI: 10.1109/TIM.2007.900132.

[MPJ13] S. Marinkovic, E. Popovici y E. Jovanov, "Improving power efficiency in WBAN communication using wake up methods", English, en *Wireless Mobile Communication and Healthcare SE - 34*, ép. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, B. Godara y K. Nikita, eds., vol. 61, Springer Berlin Heidelberg, 2013, págs. 303-317, ISBN: 978-3-642-37892-8. DOI: 10.1007/978-3-642-37893-5{_}34.

[MPS+09] S. J. Marinković, E. M. Popovici, C. Spagnol, S. Faul y W. P. Marnane, "Energy-efficient low duty cycle MAC protocol for wireless body area networks.", *IEEE transactions on information technology in biomedicine : A publication of the IEEE Engineering in Medicine and Biology Society*, vol. 13, n.º 6, págs. 915-25, nov. de 2009, ISSN: 1558-0032. DOI: 10.1109/TITB.2009.2033591.

[MPSP10] P. Muthukumaran, R. de Paz, R. Spinar y D. Pesch, "Meshmac: Enabling mesh networking over IEEE 802.15.4 through distributed beacon scheduling", English, en *Ad Hoc Networks*, ép. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, J. Zheng, S. Mao, S. Midkiff y H. Zhu, eds., vol. 28, Springer Berlin Heidelberg, 2010, págs. 561-575, ISBN: 978-3-642-11722-0. DOI: 10.1007/978-3-642-11723-7_38.

[MPT13] K. Mikhaylov, N. Plevritakis y J. Tervonen, "Performance analysis and comparison of Bluetooth low energy with IEEE 802.15.4 and SimpliciTI", *Journal of Sensor and Actuator Networks*, vol. 2, n.º 3, págs. 589-613, 2013, ISSN: 2224-2708. DOI: 10.3390/jsan2030589.

[MRM06] J. Mišić, G. R. Reddy y V. B. Mišić, "Activity scheduling based on cross-layer information in Bluetooth sensor networks", *Computer Communications*, vol. 29, n.º 17, págs. 3385-3396, nov. de 2006, ISSN: 01403664. DOI: 10.1016/j.comcom.2006.01.017.

[MSDC12] D. Miorandi, S. Sicari, F. De Pellegrini e I. Chlamtac, "Internet of things: vision, applications and research challenges", 2012. DOI: 10.1016/j.adhoc.2012.02.016.

[MSM06] J Misic, S Shafi y V. Misic, "Performance of a beacon enabled IEEE 802.15. 4 cluster with downlink and uplink traffic", *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, n.º 4, págs. 361-376, 2006, ISSN: 1045-9219. DOI: 10.1109/TPDS.2006.54.

[MT13] K. Mikhaylov y J. Tervonen, "Multihop data transfer service for Bluetooth low energy", en *2013 13th International Conference on ITS Telecommunications (ITST)*, IEEE, nov. de 2013, págs. 319-324, ISBN: 978-1-4799-0846-2. DOI: 10.1109/ITST.2013.6685566.

[MVC+15] B. Martinez, X. Vilajosana, F. Chraim, I. Vilajosana y K. S. J. Pister, "When scavengers meet industrial wireless", *IEEE Transactions on Industrial Electronics*, vol. 62, n.º 5, págs. 2994-3003, mayo de 2015, ISSN: 0278-0046. DOI: 10.1109/TIE.2014.2362891.

[MZP04] D. Miorandi, A. Zanella y G. Pierobon, "Performance evaluation of Bluetooth polling schemes: an analytical approach", en, *Mobile Networks and Applications*, vol. 9, n.º 1, págs. 63-72, 2004, ISSN: 1383469X. DOI: 10.1023/A:1027373823773.

[Mik14a] K. Mikhaylov, "Accelerated connection establishment (ACE) mechanism for Bluetooth low energy", en *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, IEEE, sep. de 2014, págs. 1264-1268, ISBN: 978-1-4799-4912-0. DOI: 10.1109/PIMRC.2014.7136362.

[Mik14b] ——, "Simulation of network-level performance for Bluetooth low energy", en *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, vol. 2015-June, IEEE, sep. de 2014, págs. 1259-1263, ISBN: 978-1-4799-4912-0. DOI: 10.1109/PIMRC.2014.7136361.

[Mis08a] J. Misic, "Analysis of slave-slave bridging in IEEE 802.15.4 beacon-enabled networks", *IEEE Transactions on Vehicular Technology*, vol. 57, n.º 3, págs. 1846-1863, mayo de 2008, ISSN: 0018-9545. DOI: 10.1109/TVT.2007.909263.

[Mis08b] J. Misic, "Traffic and energy consumption of an IEEE 802.15.4 network in the presence of authenticated, ecc diffie-hellman ephemeral key exchange", *Computer Networks*, vol. 52, n.º 11, págs. 2227-2236, 2008, ISSN: 13891286. DOI: 10.1016/j.comnet.2008.04.006.

[Mol14] D. Molloy, "Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux". Wiley, 2014.

[NBD06] L. Negri, J. Beutel y M. Dyer, "The power consumption of Bluetooth scatternets", en *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, vol. 1, IEEE, 2006, págs. 519-523, ISBN: 1-4244-0085-6. DOI: 10.1109/CCNC.2006.1593078.

[NGI+14] J Nieminen, C Gomez, M Isomaki, T Savolainen, B Patil, Z Shelby, M Xi y J Oller, "Networking solutions for connecting Bluetooth low energy enabled machines to the internet of things", *IEEE Communications and Networks*, n.º December, págs. 83-90, 2014, ISSN: 08908044. DOI: 10.1109/MNET.2014.6963809.

[NS07] B. Nefzi e Y.-Q. Song, "Performance analysis and improvement of ZigBee routing protocol", en *Fieldbuses and Networks in Industrial and Embedded Systems*, vol. 7, nov. de 2007, págs. 199-206, ISBN: 978-3-902661-34-0.

[NSI+15] J Nieminen, T Savolainen, M Isomaki, B Patil, Z Shelby y C Gomez, "IPv6 over Bluetooth low energy", RFC 7668, 2015. DOI: http://dx.doi.org/10.17487/RFC7668.

[NT06] L. Negri y L. Thiele, "Power management for Bluetooth sensor networks", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3868 LNCS, págs. 196-211, 2006, ISSN: 03029743. DOI: 10.1007/11669463_16.

[NWSS05] K. Naik, D. Wei, Y. Su y N. Shiratori, "Analysis of packet interference and aggregated throughput in a cluster of Bluetooth piconets under different traffic conditions", English, *IEEE Journal on Selected Areas in Communications*, vol. 23, n.º 6, págs. 1205-1218, jun. de 2005, ISSN: 0733-8716. DOI: 10.1109/JSAC.2005.845629.

[OROOB11] A. M. Ortiz, F. Royo, T. Olivares y L. Orozco-Barbosa, "Intelligent route discovery for ZigBee mesh networks", English, en *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, IEEE, jun. de 2011, págs. 1-6, ISBN: 978-1-4577-0352-2. DOI: 10.1109/WoWMoM.2011.5986179.

[One] "OneM2M: standards for M2M and the internet of things", ETSI, ARIB, TTA, TIA, CCSA, ATIS, TTC, TSDSI, ene. de 2015.

[Ope] "OpenMote and OpenWSN: open hardware for the internet of things", University of Berkeley, oct. de 2015.

[PAGW06] T. Pering, Y. Agarwal, R. Gupta y R. Want, "CoolSpots: reducing the power consumption of wireless mobile devices with multiple radio interfaces", en *Proceedings of the 4th international conference on Mobile systems, applications and services - MobiSys 2006*, New York, New York, USA: ACM Press, jun. de 2006, págs. 220-232, ISBN: 1595931953. DOI: 10.1145/1134680.1134704.

[PBC03] C. Petrioli, S. Basagni e I. Chlamtac, "Configuring bluestars: multihop scatternet formation for Bluetooth networks", *IEEE Transactions on Computers*, vol. 52, n.º 6, págs. 779-790, jun. de 2003, ISSN: 0018-9340. DOI: 10.1109/TC.2003.1204833.

[PBC04] C. Petrioli, S. Basagni e I. Chlamtac, "BlueMesh: degree-constrained multi-hop scatternet formation for Bluetooth networks", *Mobile Networks and Applications*, vol. 9, n.º 1, págs. 33-47, 2004, ISSN: 1383469X. DOI: 10.1023/A:1027317722864.

[PBRD03] C. Perkins, E. Belding-Royer y S. Das, "RFC 3561-ad hoc on-demand distance vector (AODV) routing", 2003.

[PEE+08] S. Pollin, M. Ergen, S. C. Ergen, B. Bougard, L. Van der Perre, I. Moerman, A. Bahai, P. Varaiya y F. Catthoor, "Performance analysis of slotted carrier sense IEEE 802.15.4 medium access layer", *IEEE Transactions on Wireless Communications*, vol. 7, n.º 9, págs. 3359-3371, 2008, ISSN: 1536-1276. DOI: 10.1109/TWC.2008.060057.

[PEFSV13] P. Park, S. C. Ergen, C. Fischione y A. Sangiovanni-Vincentelli, "Duty-cycle optimization for IEEE 802.15.4 wireless sensor networks", *ACM Transactions on Sensor Networks*, vol. 10, n.º 1, págs. 1-32, nov. de 2013, ISSN: 15504859. DOI: 10.1145/2529979.

[PK04] C. Pamuk y E. Karasan, "A tree-based energy-efficient distributed algorithm for forming bluetooth scatternet topologies", en *3rd Annual Mediterranean Ad Hoc Networking Workshop*, Bodrum, Turkey, 2004.

[PKC+05] T. Park, T. Kim, J. Choi, S. Choi y W. Kwon, "Throughput and energy consumption analysis of IEEE 802.15.4 slotted CSMA/CA", *Electronics Letters*, vol. 41, n.º 18, pág. 1017, 2005, ISSN: 00135194. DOI: 10.1049/el:20051662.

[PMS05] K. E. Persson, D. Manivannan y M. Singhal, "Bluetooth scatternets: criteria, models and classification", *Ad Hoc Networks*, vol. 3, n.º 6, págs. 777-794, 2005, ISSN: 15708705. DOI: 10.1016/j.adhoc.2004.03.014.

[PMSM15] M. Parsa, S. Motamedi, H. Safdarkhani y M. Maadani, "Performance evaluation of beacon enabled IEEE 802.15.4 network with downlink and uplink traffic and limited retransmission", *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 18, n.º 3, pág. 164, mar. de 2015, ISSN: 1743-8225. DOI: 10.1504/IJAHUC.2015.068130.

[PPMF09] G. P. Perrucci, M. V. Pedersen, T. K. Madsen y F. H. P. Fitzek, "Energy evaluation for Bluetooth link layer packet selection scheme", en *2009 European Wireless Conference, EW 2009*, 2009, págs. 250-254, ISBN: 9781424459353. DOI: 10.1109/EW.2009.5358003.

[PPV07] C. Petrioli, C. Pierascenzi y A. Vitaletti, "Bluetooth scatternet formation performance: simulations vs. testbeds", en *Proceedings - IEEE Military Communications Conference MILCOM*, 2007, ISBN: 1424406188. DOI: 10.1109/MILCOM.2006.302454.

[PRML06] M. Petrova, J. Riihijarvi, P. Mahonen y S. Labella, "Performance study of IEEE 802.15.4 using measurements and simulations", en *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.*, vol. 1, IEEE, 2006, págs. 487-492, ISBN: 1-4244-0269-7. DOI: 10.1109/WCNC.2006.1683512.

[RB08] G. R. Reddy y S. Bhatnagar, "An efficient and optimized Bluetooth scheduling algorithm for scatternets", English, en *2008 2nd International Symposium on Advanced Networks and Telecommunication Systems*, IEEE, dic. de 2008, págs. 1-3, ISBN: 978-1-4244-3600-2. DOI: 10.1109/ANTS.2008.4937783.

[RGH+09] D. Rohm, M. Goyal, H. Hosseini, A. Divjak e Y. Bashir, "A simulation based analysis of the impact of IEEE 802.15.4 MAC parameters on the performance under different traffic loads", *Mobile Information Systems*, vol. 5, n.º 1, págs. 81-99, ene. de 2009, ISSN: 1574-017X. DOI: 10.1155/2009/276978.

[RHGSGSGH13] D. Rodenas-Herraiz, A. J. Garcia-Sanchez, F. Garcia-Sanchez y J. Garcia-Haro, "Current trends in wireless mesh sensor networks: a review of competing approaches", 2013. DOI: 10.3390/s130505958.

[RKSS07] R. Roy, M. Kumar, N. K. Sharma y S. Sural, "Bottom-up construction of Bluetooth topology under a traffic-aware scheduling scheme", *IEEE Transactions on Mobile Computing*, vol. 6, n.º 1, págs. 72-86, 2007, ISSN: 15361233. DOI: 10.1109/TMC.2007.250672.

[RMHV15] S. Raza, P. Misra, Z. He y T. Voigt, "Bluetooth smart: an enabling technology for the internet of things", en *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, oct. de 2015, págs. 155-162, ISBN: 978-1-4673-7701-0. DOI: 10.1109/WiMOB.2015.7347955.

[Rf4] "RF4CE standard specification", ZigBee Alliance, 2009.

[Ria] "Red inalámbrica de comunicación de datos para un ámbito municipal", Grupo Diana, Universidad de Málaga, en colaboración con la empresa municipal de aguas de Málaga (EMASA), ene. de 2003.

[SAB+07] K. Shuaib, M. Alnuaimi, M. Boulmalf, I. Jawhar, F. Sallabi y A. Lakas, "Performance evaluation of IEEE 802.15.4: experimental and simulation results", *Journal of Communications*, vol. 2, n.º SPL.ISS. 4, págs. 29-37, 2007, ISSN: 17962021. DOI: 10.4304/jcm.2.4.29-37.

[SAKD12] S. Sharafeddine, I. Al-Kassem y Z. Dawy, "A scatternet formation algorithm for Bluetooth networks with a non-uniform distribution of devices", *Journal of Network and Computer Applications*, vol. 35, n.º 2, págs. 644-656, 2012, ISSN: 10848045. DOI: 10.1016/j.jnca.2011.10.004.

[SB11] Z Shelby y C Bormann, "6LOWPAN: The wireless embedded Internet". 2011.

[SBTL05] T. Salonidis, P. Bhagwat, L. Tassiulas y R. LaMaire, "Distributed topology construction of Bluetooth wireless personal area networks", *IEEE Journal on Selected Areas in Communications*, vol. 23, n.º 3, págs. 633-643, mar. de 2005, ISSN: 0733-8716. DOI: 10.1109/JSAC.2004.842567.

[SHB14] Z Shelby, K Hartke y C Bormann, "The constrained application protocol (CoAP)", inf. téc., 2014, pág. 112. DOI: 10.1007/s13398-014-0173-7.2.

[SHC11] B. Shrestha, E. Hossain y S. Camorlinga, "Ieee 802.15.4 mac with gts transmission for heterogeneous devices with application to wheelchair body-area sensor networks", *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, n.º 5, págs. 767-777, 2011.

[SHK12] G. Shan, X. Hao y Z. Kefeng, "The study on multipath extension of ZigBee hierarchical tree routing for wireless sensor networks", *International Journal of Digital Content Technology and its Applications*, vol. 6, n.º August, págs. 282-290, 2012, ISSN: 1975-9339. DOI: 10.4156/jdcta.vol6.issue14.35.

[SHNN12] M. Siekkinen, M. Hiienkari, J. K. Nurminen y J. Nieminen, "How low energy is Bluetooth low energy? comparative measurements with ZigBee/802.15.4", en *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, abr. de 2012, págs. 232-237, ISBN: 978-1-4673-0682-9. DOI: 10.1109/WCNCW.2012.6215496.

[SKB+14] T Savolainen, K Kerai, F Berntsen, J Decuir y R Heydon, "Internet protocol support profile", Bluetooth profile specification, dic. de 2014. DOI: http://dx.doi.org/10.17487/RFC7668.

[SKP09] S. Y. Shin, J. S. Kang y H. S. Park, "Packet error rate analysis of ZigBee under interferences of multiple Bluetooth piconets", en *IEEE Vehicular Technology Conference*, 2009, ISBN: 9781424425174. DOI: 10.1109/VETECS.2009.5073765.

[SKV03] B. Sikdar, S. Kalyanaraman y K. Vastola, "Analytic models for the latency and steady-state throughput of TCP tahoe, reno, and SACK", *IEEE/ACM Transactions on Networking*, vol. 11, n.º 6, págs. 959-971, dic. de 2003, ISSN: 1063-6692. DOI: 10.1109/TNET.2003.820427.

[SPCK07] S. Y. Shin, H. S. Park, S. Choi y W. H. Kwon, "Packet error rate analysis of ZigBee under WLAN and Bluetooth interferences", *IEEE Transactions on Wireless Communications*, vol. 6, n.º 8, págs. 2825-2830, 2007, ISSN: 15361276. DOI: 10.1109/TWC.2007.06112.

[SPK07] S. Y. Shin, H. S. Park y W. H. Kwon, "Mutual interference analysis of IEEE 802.15.4 and IEEE 802.11b", *Computer Networks*, vol. 51, n.º 12, págs. 3338-3353, ago. de 2007, ISSN: 13891286. DOI: 10.1016/j.comnet.2007.01.034.

[SS08] P. Sahoo y J.-P. Sheu, "Modeling IEEE 802.15.4 based wireless sensor network with packet retry limits", en *PE-WASUN'08: Proceedings of the 5th ACM International Symposium on Performance Evaluation of Wireless Ad-Hoc, Sensor, and Ubiquitous Networks*, 2008, págs. 63-70, ISBN: 9781605582368. DOI: 10.1145/1454609.1454624.

[SS11] F. Shu y T. Sakurai, "A new analytical model for the IEEE 802.15.4 CSMA-CA protocol", *Computer Networks*, vol. 55, n.º 11, págs. 2576-2591, ago. de 2011, ISSN: 13891286. DOI: 10.1016/j.comnet.2011.04.017.

[SSG07] M. Song, S. Shetty y D. Gopalpet, "Coexistence of IEEE 802.11b and Bluetooth: an integrated performance analysis", en, *Mobile Networks and Applications*, vol. 12, n.º 5-6, págs. 450-459, abr. de 2007, ISSN: 1383469X. DOI: 10.1007/s11036-008-0047-3.

[SVW+14] D. Stanislowski, X. Vilajosana, Q. Wang, T. Watteyne y K. S. J. Pister, "Adaptive synchronization in IEEE 802.15.4e networks", *IEEE Transactions on Industrial Informatics*, vol. 10, n.º 1, págs. 795-802, feb. de 2014, ISSN: 1551-3203. DOI: 10.1109/TII.2013.2255062.

[SXZ15] C.-F. Shih, A. E. Xhafa y J. Zhou, "Practical frequency hopping sequence design for interference avoidance in 802.15.4e TSCH networks", en *2015 IEEE International Conference on Communications (ICC)*, IEEE, jun. de 2015, págs. 6494-6499, ISBN: 978-1-4673-6432-4. DOI: 10.1109/ICC.2015.7249359.

[SZ06] I. Stojmenovic y N. Zaguia, "Bluetooth scatternet formation in ad hoc wireless networks", *Performance Modeling and Analysis of Bluetooth Networks: Network Formation, Polling, Scheduling, and Traffic Control*, 2006.

[SZR+09] Z. Sun, X.-G. Zhang, D. Ruan, H. Li y X. Pang, "A routing protocol based on flooding and AODV in the ZigBee network", English, en *2009 International Workshop on Intelligent Systems and Applications*, IEEE, mayo de 2009, págs. 1-4, ISBN: 978-1-4244-3893-8. DOI: 10.1109/IWISA.2009.5072672.

[Sar13] P. P. Saraswala, "A survey on routing protocols in ZigBee networks", *International Journal of Engineering Science and Innovative Technology (IJESIT)*, vol. 2, n.º 1, 2013.

[Sim] "Simpliciti overview", Texas Instruments, 2008.

[TM06] Tae-Jin Lee y Min Young Chung, "Communications letters, ieee", 2006.

[TMPS11] M. Tariq, M. Macuha, Y.-J. Park y T. Sato, "Performance evaluation of the IEEE 802.15.4 multi-hop communications in error-prone wireless sensor networks", en *Proceedings of the 9th ACM international symposium on Mobility management and wireless access - MobiWac '11*, New York, New York, USA: ACM Press, oct. de 2011, pág. 131, ISBN: 9781450309011. DOI: 10.1145/2069131.2069154.

[TS04] N. Timmons y W. Scanlon, "Analysis of the performance of IEEE 802.15.4 for medical sensor body area networking", en *Proceeding of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004.*, IEEE, 2004, págs. 16-24, ISBN: 0-7803-8796-1. DOI: 10.1109/SAHCN.2004.1381898.

[TSJ+14] Taehong Kim, Seong Hoon Kim, Jinyoung Yang, Seong-eun Yoo y Daeyoung Kim, "Neighbor table based shortcut tree routing in ZigBee wireless networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, n.º 3, págs. 706-716, mar. de 2014, ISSN: 1045-9219. DOI: 10.1109/TPDS.2014.9.

[TSPB15] J. J. Treurniet, C. Sarkar, R. V. Prasad y W. de Boer, "Energy consumption and latency in BLE devices under mutual interference: an experimental study", English, en *2015 3rd International Conference on Future Internet of Things and Cloud*, IEEE, ago. de 2015, págs. 333-340, ISBN: 978-1-4673-8103-1. DOI: 10.1109/FiCloud.2015.108.

[TWP13] E. Tabatabaei Yazdi, A. Willig y K. Pawlikowski, "Shortening orphan time in IEEE 802.15.4: what can be gained?", en *2013 19th IEEE International Conference on Networks (ICON)*, IEEE, dic. de 2013, págs. 1-6, ISBN: 978-1-4799-2084-6. DOI: 10.1109/ICON.2013.6781984.

[Tan11] J. Tang, "A policy of reducing power consumption for Bluetooth devices", en *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, vol. 2, IEEE, sep. de 2011, págs. 145-148, ISBN: 978-1-4577-1419-1. DOI: 10.1109/ICM.2011.57.

[VTPVG+14] X. Vilajosana, P. Tuset-Peiro, F. Vazquez-Gallego, J. Alonso-Zarate y L. Alonso, "Standardized low-power wireless communication technologies for distributed sensing applications.", en, *Sensors (Basel, Switzerland)*, vol. 14, n.º 2, págs. 2663-82, ene. de 2014, ISSN: 1424-8220. DOI: 10.3390/s140202663.

[VWC+14] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang y K. S. J. Pister, "A realistic energy consumption model for TSCH networks", *IEEE Sensors Journal*, vol. 14, n.º 2, págs. 482-489, feb. de 2014, ISSN: 1530-437X. DOI: 10.1109/JSEN.2013.2285411.

[WGL12] S. Wijetunge, U. Gunawardana y R. Liyanapathirana, "Performance analysis of IEEE 802.15.4 MAC protocol with ACK frame transmission", *Wireless Personal Communications*, vol. 69, n.º 2, págs. 509-534, abr. de 2012, ISSN: 0929-6212. DOI: 10.1007/s11277-012-0587-5.

[WGL14] ——, "Throughput analysis of IEEE 802.15.4 beacon-enabled MAC protocol in the presence of hidden nodes", *Wireless Networks*, vol. 20, n.º 7, págs. 1889-1908, abr. de 2014, ISSN: 1022-0038. DOI: 10.1007/s11276-013-0637-2.

[WHC05] R. M. Whitaker, L. Hodge e I. Chlamtac, "Bluetooth scatternet formation: a survey", *Ad Hoc Networks*, vol. 3, n.º 4, págs. 403-450, 2005, ISSN: 15708705. DOI: 10.1016/j.adhoc.2004.02.002.

[WN13] J. Wen y J. Nelson, "Modelling and performance analysis of an adaptive state-transition approach for power saving in Bluetooth", *Simulation Modelling Practice and Theory*, vol. 31, págs. 77-95, feb. de 2013, ISSN: 1569190X. DOI: 10.1016/j.simpat.2012.11.005.

[WSJ15] R. Want, B. N. Schilit y S. Jenson, "Enabling the internet of things", *Computer*, vol. 48, n.º 1, págs. 28-35, ene. de 2015, ISSN: 0018-9162. DOI: 10.1109/MC.2015.12.

[WSL13] R. Want, B. Schilit y D. Laskowski, "Bluetooth LE finds its niche", *IEEE Pervasive Computing*, vol. 12, n.º 4, págs. 12-16, 2013, ISSN: 15361268. DOI: 10.1109/MPRV.2013.60.

[WTB+12] T. Winter, P. Thubert, A. Brandt, T. H. Clausen, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik y J. Vasseur, "RPL: IPv6 routing protocol for low-power and lossy networks", RFC 6550, 2012. DOI: 10.2313/NET-2011-07-1.

[WW08a] E. U. Warriach y S. Witte, "Approach for performance investigation of different Bluetooth modules and communication modes", en *Proceedings - 4th IEEE International Conference on Emerging Technologies 2008, ICET 2008*, 2008, págs. 167-171, ISBN: 9781424422111. DOI: 10.1109/ICET.2008.4777494.

[WW08b] W. T. Woon y T.-C. Wan, "Performance evaluation of IEEE 802.15.4 wireless multi-hop networks: simulation and testbed approach", *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 3, n.º 1, págs. 57-66, dic. de 2008, ISSN: 1743-8225. DOI: 10.1504/IJAHUC.2008.016195.

[WXLC13] H. Wang, M. Xi, J. Liu y C. Chen, "Transmitting IPv6 packets over Bluetooth low energy based on BlueZ", en *Advanced Communication Technology (ICACT), 2013 15th International Conference on*, ene. de 2013, págs. 72-77.

[Wei14] J. Wei, "How wearables intersect with the cloud and the internet of things: considerations for the developers of wearables.", *IEEE Consumer Electronics Magazine*, vol. 3, n.º 3, págs. 53-56, jul. de 2014, ISSN: 2162-2248. DOI: 10.1109/MCE.2014.2317895.

[YCSK08] D. S. Yun, S. H. Cho, D. W. Seo y M. S. Kang, "An efficient and reliable data transmission control method for relaxing congestion problem in ZigBee network", en *Proceedings of the 2nd international conference on Ubiquitous information management and communication - ICUIMC '08*, New York, New York, USA: ACM Press, ene. de 2008, pág. 533, ISBN: 9781595939937. DOI: 10.1145/1352793.1352905.

[YP14] L. W. Yeh y M. S. Pan, "Beacon scheduling for broadcast and convergecast in ZigBee wireless sensor networks", *Computer Communications*, vol. 38, págs. 1-12, 2014, ISSN: 01403664. DOI: 10.1016/j.comcom.2013.10.009.

[YPGS07] Z. Yijin, X. Pingping, B. Guangguo y F Sheng Bao, "Analysis of energy efficiency and power saving in IEEE 802.15.4", en *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, IEEE, 2007, págs. 3330-3334, ISBN: 1525-3511. DOI: 10.1109/WCNC.2007.613.

[YTD+03] Yang-Ick Joo, Tae-Jin Lee, Doo Seop Eom, Yeonwoo Lee y Kyun Hyon Tchah, "Power-efficient and QoS-aware scheduling in Bluetooth scatternet for wireless PANs", *IEEE Transactions on Consumer Electronics*, vol. 49, n.º 4, págs. 1067-1072, nov. de 2003, ISSN: 0098-3063. DOI: 10.1109/TCE.2003.1261197.

[YZJ07] L. Yan, L. Zhong y N. Jha, "Energy comparison and optimization of wireless body-area network technologies", en *Proceedings of the ICST 2nd international conference on Body area networks*, ép. BodyNets '07, ICST, Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics y Telecommunications Engineering), 2007, págs. 1-8, ISBN: 978-963-06-2193-9. DOI: 10.4108/bodynets.2007.150.

[ZBC+14] A. Zanella, N. Bui, A. Castellani, L. Vangelista y M. Zorzi, "Internet of things for smart cities", *IEEE Internet of Things Jour-*

nal, vol. 1, n.º 1, págs. 22-32, feb. de 2014, ISSN: 2327-4662. DOI: 10.1109/JIOT.2014.2306328.

[ZBC01] G. Zaruba, S. Basagni e I. Chlamtac, "Bluetrees-scatternet formation to enable bluetooth-based ad hoc networks", *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, vol. 1, 2001, ISSN: 05361486. DOI: 10.1109/ICC.2001.936316.

[ZL04] J. Z. J. Zheng y M. Lee, "Will IEEE 802.15.4 make ubiquitous networking a reality?: a discussion on a potential low power, low bit rate standard", English, *IEEE Communications Magazine*, vol. 42, n.º 6, págs. 140-146, jun. de 2004, ISSN: 0163-6804. DOI: 10.1109/MCOM.2004.1304251.

[ZL06] J. Zheng y M. J. Lee, "A comprehensive performance study of IEEE 802.15. 4", en *Sensor network operations*, vol. 4, 2006, págs. 1-14, ISBN: 0780392833. DOI: 10.1109/ICICS.2005.1689245.

[ZSD08] N. Zaguia, I. Stojmenovic e Y. Daadaa, "Simplified Bluetooth scatternet formation using maximal independent sets", en *2008 International Conference on Complex, Intelligent and Software Intensive Systems*, IEEE, 2008, págs. 443-448, ISBN: 978-0-7695-3109-0. DOI: 10.1109/CISIS.2008.16.

[ZSY07] G. Zussman, A. Segall y U. Yechiali, "On the analysis of the Bluetooth time division duplex mechanism", *IEEE Transactions on Wireless Communications*, vol. 6, n.º 6, págs. 2149-2160, 2007, ISSN: 15361276. DOI: 10.1109/TWC.2007.05152.

[ZWDL08] F. Zhang, F. Wang, B. Dai e Y. Li, "Performance evaluation of IEEE 802.15.4 beacon-enabled association process", en *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, IEEE, 2008, págs. 541-546, ISBN: 978-0-7695-3096-3. DOI: 10.1109/WAINA.2008.241.

[ZZ08] A. Zanella y M. Zorzi, "Throughput and energy efficiency of Bluetooth v2+EDR in fading channels", en *2008 IEEE Wireless Communications and Networking Conference*, IEEE, mar. de 2008, págs. 1661-1666, ISBN: 978-1-4244-1997-5. DOI: 10.1109/WCNC.2008.297.

[ZZZ09] F. Z. F. Zhang, H. Z. H. Zhou y X. Z. X. Zhou, "A routing algorithm for ZigBee network based on dynamic energy consumption decisive path", *2009 International Conference on Computational Intelligence and Natural Computing*, vol. 1, págs. 429-432, 2009. DOI: 10.1109/CINC.2009.193.

[Zan09a] A. Zanella, "A mathematical framework for the performance analysis of Bluetooth with enhanced data rate", *IEEE Transactions on Communications*, vol. 57, n.º 8, págs. 2463-2473, ago. de 2009, ISSN: 0090-6778. DOI: 10.1109/TCOMM.2009.08.070662.

[Zan09b] ——, "A mathematical framework for the performance analysis of Bluetooth with enhanced data rate", *IEEE Transactions on Communications*, vol. 57, n.º 8, págs. 2463-2473, ago. de 2009, ISSN: 0090-6778. DOI: 10.1109/TCOMM.2009.08.070662.

[Zbsa] "ZigBee 2012 specifications", ZigBee Alliance, 2006.

[Zbsb] "ZigBee specifications", ZigBee Alliance, 2006.

[Zst] "Z-Stack: Fully compliant ZigBee protocol solution", Texas Instruments, 2010.

[Cro09] CrossBow Technologies, Inc, "TelosB mote platform", 2009.

[Sil15] C. Silvent Aparicio, "Z-Wave PHY/MAC evaluation: new standard for the internet of things", eng, Master Thesis, UPC, nov. de 2015.

[ÖDE+06] F. Österlind, A. Dunkels, J. Eriksson, N. Finne y T. Voigt, "Cross-level sensor network simulation with COOJA", en *Proceedings - Conference on Local Computer Networks, LCN*, 2006, págs. 641-648, ISBN: 1424404185. DOI: 10.1109/LCN.2006.322172.