UNIVERSIDAD DE DEUSTO

MEJORA DE LA DISPONIBILIDAD DE SUBREDES PRIME DURANTE EL PROCESO DE ACTUALIZACIÓN FIRMWARE

Tesis doctoral presentada por Janire Larrañaga Arriola
dentro del Programa de Doctorado en
Ingeniería Informática y Telecomunicación
Dirigida por el Dr. Jon Legarda Macón

Bilbao, diciembre de 2015

UNIVERSIDAD DE DEUSTO

MEJORA DE LA DISPONIBILIDAD DE SUBREDES PRIME DURANTE EL PROCESO DE ACTUALIZACIÓN FIRMWARE

Tesis doctoral presentada por Janire Larrañaga Arriola
dentro del Programa de Doctorado en
Ingeniería Informática y Telecomunicación

Dirigida por el Dr. Jon Legarda Macón

La doctoranda El director

Bilbao, diciembre de 2015

Mejora de la disponibilidad de subredes PRIME durante el proceso de actualización firmware

Autora: Janire Larrañaga Arriola
Director: Dr. Jon Legarda Macón

Editado en LaTeX2e utilizando *TeX Gyre Heros* para títulos y *Charter BT* para el contenido general.

Impreso en Bilbao
Primera edición, diciembre de 2015

Nire gurasoei, Jon Koldo eta Josune,
eta aitxitxe Juan eta amuma Bittoriri
bihotz-bihotzez

Resumen

La tecnología *Power Line Communications (PLC)* desempeña un papel clave en la evolución de las redes eléctricas hacia las *Smart Grid*s como lo demuestra la tecnología *PowerLine Intelligent Metering Evolution (PRIME)*, el estándar que actualmente soportan los contadores inteligentes desplegados en España y otros países de Europa.

Actualmente, PRIME cuenta con las prestaciones adecuadas para soportar aplicaciones de telemedida, pero con la llegada de las *Smart Grid*s surgirán nuevos retos tecnológicos que requerirán la mejora de sus prestaciones. Las operadoras eléctricas muestran la necesidad de mejorar el rendimiento de las aplicaciones que actualmente más recursos demandan, como por ejemplo el proceso de actualización *firmware* de los equipos de telemedida de las redes PRIME.

Este trabajo de investigación pone su atención en la disponibilidad de red como un recurso crítico de las *Smart Grid* y tiene como objetivo mejorar la disponibilidad de la red durante el proceso de actualización *firmware*, sin que a cambio suponga un aumento de la duración del proceso. Actualmente, no existe un consenso sobre qué estrategias de actualización *firmware* son las más eficientes en estos términos, ya que aunque la aplicación de actualización *firmware* es una aplicación definida en gran medida por el estándar PRIME muchos de los aspectos clave son de libre implementación, lo que al mismo tiempo proporciona cierto margen de mejora. Este objetivo ha llevado a plantear la hipótesis de si es posible mejorar la disponibilidad de las subredes PRIME durante el proceso de actualización *firmware* sin aumentar la duración total del proceso mediante el diseño de estrategias de actualización que tomen en cuenta la topología lógica de la subred.

Con este objetivo en mente, este trabajo de investigación cuenta con dos contribuciones principales. Por un lado, se ha realizado un estudio que incluye el diseño, implementación y evaluación de distintas estrategias de actualización *firmware* que toman en cuenta la topología lógica de la red y han sido evaluadas en términos de disponibilidad de subred y tiempo de actualización sobre escenarios

de distinta topología física y lógica. Por otro lado, para realizar dicho estudio, teniendo en cuenta que realizar estudios exhaustivos de estas características sobre redes reales es impracticable, se ha desarrollado un modelo de simulación de subredes PRIME que ha sido validado con resultados de pruebas sobre redes reales.

Los resultados demuestran que la topología lógica de la red es efectivamente un aspecto clave a considerar en el diseño de estrategias de actualización *firmware*. A su vez, se propone una estrategia de actualización que mejora claramente la disponibilidad de subred especialmente en los escenarios más profundos, sin aumentar la duración del proceso.

Agradecimientos

En primer lugar, me gustaría agradecer a mi director de tesis Jon Legarda por la confianza depositada en mí para realizar este trabajo de investigación, así como por su paciencia y dedicación a lo largo de estos años.

Agradezco también a Alberto Sendín y a Iker Urrutia, miembros del departamento de telecomunicaciones de Iberdrola, por sus valiosos aportes y sugerencias realizadas.

No puedo olvidarme de mis compañeros y ex-compañeros de trabajo del grupo Morelab, con los que además de trabajar durante todos estos años en la realización de distintos proyectos de investigación y aprender infinidad de cosas de ellos, he tenido la oportunidad compartir viajes, celebraciones, escapadas a la nieve, etc. Seguid siendo como sois.

He tenido la suerte de contar con muy buenos compañeros de trabajo, pero me gustaría agradecer de manera especial a Unai Aguilera por sus valiosos consejos y esas contínuas inyecciones de ánimo que tanto me han ayudado a seguir adelante en este duro camino. En definitiva, gracias por esas largas conversaciones y por ayudarme a no tirar la toalla, haciéndome ver lo mucho que merecía la pena el esfuerzo. A Iván Pretel, gran amigo, compañero de trabajo y de travesía durante la realización de la tesis doctoral. Gracias también por tus consejos y por los buenos ratos que hemos pasado tanto en el laboratorio como fuera de él. A Koldo Zabaleta gran compañero de trabajo y mejor amigo, siempre dispuesto a escuchar y a hacer un “break” cuando más lo he necesitado.

Por otro lado, no quisiera dejar de agradecer a personas que son importantes para mí, que aunque no están relacionados con el mundo de la ciencia, son también responsables de la consecución de esta tesis doctoral.

A Ainhoa Vicarregui con la que he compartido momentos increibles y que siempre ha estado ahí cuando más la he necesitado. Gracias por todos esos momentos y por seguir siendo mi amiga a pesar de las todas las horas de compañía que te debo y que a partir de ahora

espero que podamos recuperar. A Dámaris Merlo, por su apoyo y paciencia durante todo este tiempo. Gracias también por seguir ahí, y espero que a partir de ahora podamos llevar a cabo todos esos planes que tenemos pendientes.

Por último, agradecer a mis padres, a mis abuelos y a mi mascota Boly por el apoyo recibido durante los momentos más complicados de la investigación ayundándome a "recargar pilas" cuando más lo necesitaba. No hubiera podido conseguirlo sin ellos.

Eskerrik asko,

Janire Larrañaga

Bilbao, diciembre de 2015

Tabla de contenidos

Lista de figuras

Lista de tablas

Acrónimos

ADP	Adaptation Layer.
AIEE	American Institute of Electrical Engineers.
AMI	Advanced Metering Infrastructure.
AMR	Automatic Meter Reading.
ANSI	American National Standards Institute.
API	Application Programming Interface.
AT	Alta Tensión.
ATM	Adaptive Tone Mapping.
BER	Bit Error Rate.
BN	Base Node.
BPDU	Beacon PDU.
BPSK	Binary Phase Shift Keying.
BSI	Beacon Slot Indication.
BT	Baja Tensión.
CA	Corriente Alterna.
CC	Convolutional Code.
CENELEC	Comité Européen de Normalisation Electrotechnique.
CFP	Contention Free Period.
CID	Connection Identifier.
CL	Convergence Layer.
CPCS	Common Part Convergence Layer.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Code.
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance.
CSP	Clustered Simple Polling.
CWDM	Coarse Wavelength Division Multiplex.
D8PSK	Differential 8 Phased Shift Keying.
DBPSK	Differential Binary Phased Shift Keying.
DLL	Data Link Layer.

DLMS/COSEM	Device Language Message Specification/Companion Specification for Energy Metering.
DQPSK	Differential Quadrature Phased Shift Keying.
DWDM	Dense Wavelength Division Multiplex.
ECSS	Embedded Communications Software Stack.
FCC	Federal Communications Commission.
FEC	Forward Error Correction.
FER	Frame Error Rate.
FFT	Fast Fourier Transform.
FIR	Finite Impulse Response.
FSK	Frequency Shift Keying.
GAD	Gestión Activa de la Demanda.
GPDU	Generic MAC Protocol Data Unit.
GUI	Graphical User Interface.
HAN	Home Area Network.
IEEE	Institute of Electrical and Electronics Engineers.
IPC	Inter Process Communication.
ISP	Inter-System Protocol.
ISPLC	International Symposium on Power Line Communications and its Applications.
ITU	International Telecommunication Union.
LCID	Local Connection Identifier.
LDPC	Low Density Parity Check.
LLC	Logical Link Control.
LNID	Local Node Identifier.
LSID	Local Switch Identifier.
LV	Low Voltage.
MAC	Media Access Layer.
MPDU	MAC Protocol Data Unit.
MPI	Message Passing Interface.
MSDU	MAC Service Data Unit.
MT	Media Tensión.
NED	Network Description.
NID	Node Identifier.
NRP	Neighbour Rellay Polling.
OFDM	Orthogonal Frequency Division Multiplexing.
PDH	Plesiochronous Digital Hierarchy.
PDU	Protocol Data Unit.
PHY	Physical Layer.
PIB	PLC Information Base.
PLC	Power Line Communications.

PNPDU	Promotion Needed PDU.
PPDU	PHY Protocol Data Unit.
PRIME	PowerLine Intelligent Metering Evolution.
QAM	Quadrature Amplitude Modulation.
QPSK	Quaternary Phase Shift Keying.
RF	Radio frequency.
RS	Reed Solomon.
SCP	Shared Contention Period.
SDH	Synchronous Digital Hierarchy.
SID	Switch Identifier.
SNA	Subnetwork Address.
SNR	Signal-to-Noise Ratio.
SQL	Structured Query Language.
SSCS	Service Specific Convergence Sublayer.
STM	Synchronous Transport Module.
Tcl	Tool Command Language.
TIA	Telecommunications Industry Association.
UART	Universal Asynchronous Receiver-Transceiver.
WAN	Wide Area Network.
WPAN	Wireless Personal Area Network.
XML	eXtensible Markup Language.

Capítulo

1

Introducción

El sector energético se enfrenta a importantes retos tecnológicos en el proceso de transformación del modelo productivo hacia formas más sostenibles de producción y consumo. La estrategia europea actual pretende alcanzar un 20 % de energías de origen renovable, un 20 % de reducción de las emisiones de gases de efecto invernadero y un 20 % de incremento de la eficiencia energética [Eur10]. La piedra angular para el cumplimiento de estos compromisos y estos desafíos tecnológicos de baja emisión de carbono es la red eléctrica inteligente. La red inteligente o *Smart Grid* es una evolución tecnológica del sistema de distribución de energía que combina las instalaciones eléctricas tradicionales con modernas tecnologías de monitorización, sistemas de información y telecomunicaciones. Permitirá ofrecer un amplio abanico de servicios a los clientes, mejorar la calidad de suministro, atender las necesidades en términos de energía eléctrica que va a demandar la sociedad en el futuro y gestionar la distribución de energía de forma óptima. Se trata de una red que integra de manera inteligente las acciones de los usuarios que se encuentran conectados a ella: generadores, consumidores y aquéllos que son ambas cosas a la vez, con el fin de conseguir un suministro eléctrico eficiente, seguro y sostenible.

Las redes inteligentes o *Smart Grid*s utilizarán equipos y servicios innovadores, junto con nuevas tecnologías de comunicación, control, monitorización y auto-diagnóstico que ayudarán a conseguir los siguientes objetivos [UPC15]:

- Robustecer y automatizar la red, mejorando la operación de la red, la calidad de suministro y minimizar las pérdidas en la misma.
- Optimizar la conexión de las zonas con fuentes de energía renovable, optimizando las capacidades de conexión y minimizando el coste de conexión de las mismas.

- Desarrollar arquitecturas de generación descentralizadas, permitiendo el funcionamiento de instalaciones de menor tamaño (Generación Distribuida) en armonía con el sistema.

- Mejorar la integración de la generación intermitente y de nuevas tecnologías de almacenamiento.

- Avanzar en el desarrollo del mercado de la electricidad, posibilitando nuevas funcionalidades y servicios a los comercializadores y a millones de consumidores en el mercado.

- Gestionar de forma activa la demanda, permitiendo que los consumidores gestionen de manera más eficiente sus consumos, mejorando así la eficiencia energética.

- Posibilitar la penetración del vehículo eléctrico, acomodando estas nuevas cargas móviles y dispersas a la red, minimizando el desarrollo de nueva infraestructura y habilitando las funcionalidades de almacenamiento de energía que poseen.

El primer paso hacia las *Smart Grids* son los sistemas de medición inteligente o *Smart Metering*, también conocidos como sistemas de *Automatic Meter Reading (AMR)*. El *Smart Metering* proporciona no sólo la capacidad de realizar operaciones de medición remotas, sino que también permite operar directamente sobre los contadores (conexión/desconexión). Por otra parte, los contadores inteligentes proporcionan información muy importante acerca de la red: carga en el transformador, consumo de electricidad en cada alimentador de baja tensión, patrones históricos de consumo, etc. Esta valiosa información permite incrementar el control global de la red eléctrica y optimiza la gestión de consumo de energía, por ejemplo, a través de tarifas basadas en el conocimiento del consumo en tiempo real y limitando el consumo máximo de energía en ciertos momentos.

Los programas de medición inteligente se han convertido en una realidad en todo el mundo y su despliegue se ve fomentado tanto por el interés que tienen las compañías eléctricas como por las leyes o mandatos gubernamentales. En concreto, en España, en junio de 2006 se aprobó el Real Decreto 809/2006 [RD806] en el que se establecía que a partir del 1 de julio de 2007 los equipos de medida a instalar para los nuevos suministros y los que sustituyeran a los antiguos deberán permitir la discriminación horaria en las medidas así como la telegestión. Asimismo, todos los contadores de medida en suministros de energía eléctrica con una potencia contratada de hasta 15 kW deberán ser sustituidos por los nuevos equipos antes del 31 de diciembre de 2018 [ORD07].

Power Line Communications (PLC) se perfila como una de las principales tecnologías para proporcionar comunicaciones a los sistemas de distribución de

energía eléctrica [DBS11, PVYH03, GSW11, SBA+13], y por lo tanto, para implementar los sistemas de *Smart Metering* así como los futuros servicios y aplicaciones de las *Smart Grids*. Esta tecnología aprovecha el cableado eléctrico existente para transmitir señales digitales y proveer distintos servicios de telecomunicación a través de la infraestructura existente, reduciendo así el coste de su instalación. Por esta razón, debido a su integración con la red eléctrica, la tecnología PLC tiene una larga tradición entre las compañías eléctricas. Un ejemplo de este tipo de tecnología es *PowerLine Intelligent Metering Evolution (PRIME)* [PRI08], un estándar PLC mundialmente consolidado que está siendo desplegado en España y otros países de Europa por la operadora eléctrica Iberdrola [Ber08, ABS+10, SLAB11a, SBA+12, SLAB11b].

PRIME ofrece buenos resultados en las operaciones de *metering* o medición que se realizan actualmente [ABS+10, SLAB11a, SBA+12], pero los futuros servicios o aplicaciones que se pretenden proveer a través de este tipo de redes traerán nuevos retos tecnológicos que exigirán una mejora de sus prestaciones.

Algunos de estos retos surgirán del elevado volumen de datos que se va a generar debido al crecimiento exponencial de los dispositivos inteligentes desplegados en la red. Como ya se ha adelantado, los dispositivos inteligentes proporcionarán información muy importante acerca de la red (patrones de consumo de los usuarios, la carga en el transformador, etc.) que permitirá aumentar el control sobre la red y mejorar la gestión del consumo de energía. Este crecimiento del volumen datos requerirá un mayor almacenamiento y ancho de banda en las comunicaciones.

La carga inteligente de vehículos eléctricos también requerirá una mejora de las prestaciones en términos ancho de banda a raíz del tráfico generado en las funciones de control y monitorización del estado de carga de los vehículos y la optimización de dicha carga.

Por otro lado, la seguridad también será un aspecto clave en las redes de comunicaciones de las *Smart Grids* ya que son vulnerables a diversos ataques de seguridad, como por ejemplo los ataques de denegación de servicio o de suplantación de identidad. La seguridad de las comunicaciones está a su vez relacionada con la privacidad de los consumidores, y la cuestión de quién tiene los datos y por qué motivo es una cuestión que preocupa a los legisladores actualmente. La información procedente de contadores inteligentes puede ser combinada en formas inesperadas y revelar información que los consumidores no desean que se conozca.

Por último, surgirán aplicaciones que requerirán realizar acciones críticas en tiempo real como por ejemplo, la localización de fallos en la red, la desconexión de los usuarios que no pagan, la desconexión de partes de la red en caso de peligro, etc. [FLNS10]. Todas estas acciones requerirán que la disponibilidad de las comunicaciones sea lo más alta posible.

1.1 Motivación

Tal y como se ha visto en el apartado anterior, las *Smart Grids* se enfrentan a multitud de retos tecnológicos en los próximos años. Uno de esos retos reside en el hecho de que las tecnologías de telecomunicación deben adecuar las prestaciones que ofrecen actualmente (en términos de ancho de banda, latencia, seguridad, disponibilidad, etc.) a los nuevos escenarios, servicios o aplicaciones que irá demandando el sector. Ante este nuevo escenario, las operadoras eléctricas como Iberdrola muestran la necesidad de mejorar el rendimiento de las aplicaciones que actualmente más recursos demandan de las redes de telecomunicaciones.

Este trabajo, por un lado, pone su atención en la disponibilidad de red como un recurso crítico de las *Smart Grid*, entendiendo como disponibilidad el intervalo de tiempo durante el cual un dispositivo puede comunicarse con otros dispositivos de la red con respecto a un tiempo total de observación, y como disponibilidad de red a la media de disponibilidades de todos los dispositivos que la componen [SBA+13]. Y por otro lado, este trabajo adopta como caso de aplicación real la actualización *firmware* de los equipos de telemedida, una aplicación que ocupa el canal de comunicaciones durante un tiempo elevado, si lo comparamos con el necesario para hacer una simple lectura, y además inutiliza los equipos de telemedida durante el proceso de reinicio necesario para la actualización, lo que implica inutilizar todos sus equipos dependientes.

En este contexto, el estado del arte no ofrece un consenso sobre qué estrategias de actualización *firmware* son las más eficientes, y menos en términos de disponibilidad de red o duración del proceso. Así, este trabajo contribuye con un estudio que incluye el diseño, implementación y evaluación de distintas estrategias de actualización *firmware* con el objetivo de mejorar la disponibilidad sin aumentar el tiempo total del proceso. Esto es posible gracias a que, tal y como se verá a lo largo del documento, la aplicación de actualización *firmware* es una aplicación definida en gran medida por el estándar PRIME pero a la vez indefinida en muchos de los aspectos clave de su implementación.

Por otro lado, realizar estudios exhaustivos de estas características sobre redes reales es impracticable, debido a que el acceso a despliegues reales es complicado y a que el tiempo necesario para realizarlos es inasumible. Este hecho hace que sea necesario recurrir a herramientas de simulación que permitan hacer estudios de forma sistemática. En la actualidad el estado del arte ofrece una serie de simuladores que, como se verá a lo largo del documento, o bien no están validados en escenarios reales o la metodología necesaria para reproducir el comportamiento de las redes reales es muy costosa (situando *sniffers* en distintos puntos de la red) y requiere de información fiable de la topología física de las redes que no es 100 % conocida por las compañías eléctricas [SBA+13]. Este hecho ha motivado el diseño, desarrollo y validación de una herramienta

de simulación de subredes PRIME que será a su vez otra de las contribuciones de este trabajo de investigación.

1.2 Hipótesis y objetivos

De acuerdo a lo que se ha explicado en los apartados anteriores, el ***objetivo general*** de este trabajo de investigación es *mejorar la disponibilidad de las subredes PRIME durante el proceso de actualización firmware, sin que a cambio suponga un aumento del tiempo necesario para la actualización de los nodos de servicio de la red*. La ***hipótesis de partida*** que se ha formulado en base a este objetivo se presenta a continuación.

> *La disponibilidad de las subredes PRIME durante el proceso de actualización firmware de los nodos de servicio que componen dicha subred puede ser mejorada sin aumentar la duración total del proceso mediante el diseño de estrategias de actualización que tomen en cuenta la topología lógica de la subred.*

Para alcanzar el objetivo general planteado, se definen varios ***objetivos específicos***. A su vez, para la consecución de dichos objetivos específicos se expondrán los ***objetivos operacionales***.

- ***O.E.1:*** *Desarrollar y validar un modelo de simulación de subredes PRIME que permita evaluar la aplicación de actualización firmware.* Para poder llevar a cabo este trabajo investigación se ve necesario el desarrollo de un modelo de simulación de subredes PRIME propio, ya que no se dispone de un simulador este tipo y la realización pruebas en campo requiere de mucho tiempo y esfuerzo.
  - ***O.O.1:*** *Diseñar la arquitectura del modelo de simulación.* En primer lugar, se definirán los distintos tipos de módulos que compondrán el modelo de simulación así como su estructura y funcionalidad. Por otra parte, también se definirán otros aspectos como los parámetros de entrada y salida del modelo así como su formato.
  - ***O.O.2:*** *Implementar el modelo de simulación.* La implementación del modelo de simulación se realizará de acuerdo al diseño realizado y siguiendo la especificación 1.3.6 del estándar PRIME.
  - ***O.O.3:*** *Corregir errores y añadir mejoras al modelo de simulación.* Para verificar que el modelo de simulación funciona según lo esperado, se comenzará con la simulación de escenarios más sencillos, y con el apoyo del entorno gráfico se irán corrigiendo los errores que puedan

surgir. Poco a poco se irá incrementando la complejidad de los escenarios hasta haber depurado el código por completo. Llegados a este punto, la simulaciones se podrán comenzar a ejecutar de forma rápida y sin la supervisión del usuario mediante una interfaz de usuario simple que soporte la ejecución por lotes.

  - ***O.O.4:*** *Validar el modelo de simulación.* La validación es el proceso mediante el cual se comprueba que la solución implementada es correcta desde un punto de vista funcional. En este proceso de validación, además de realizar dicha comprobación, los resultados proporcionados por las simulaciones se compararán con los de pruebas reales, de tal forma que se pueda verificar que el modelo es capaz de reproducir el comportamiento de redes reales.

- ***O.E.2:*** Diseñar, implementar y evaluar distintas estrategias de actualización *firmware* que mejoren la disponibilidad de la subred, sin aumentar la duración total del proceso.

  - ***O.O.5:*** *Diseñar las estrategias de actualización firmware de equipos PRIME.* Las estrategias de actualización se diseñarán aprovechando los aspectos de libre implementación del proceso de actualización descrito por el estándar PRIME. Teniendo en cuenta la estructura de árbol característica de las redes PRIME, para validar la hipótesis planteada se diseñarán distintas estrategias de actualización *firmware*, entre las que se encuentran aquellas que toman en cuenta la topología lógica de la red y las que no.

  - ***O.O.6:*** *Implementar las estrategias de actualización firmware de equipos PRIME.* Una vez diseñadas las distintas estrategias de actualización se procederá a su implementación en el modelo de simulación desarrollado.

  - ***O.O.7:*** *Definir los escenarios de pruebas.* Se definirán redes de distintas topologías (físicas y lógicas) que permitan evaluar el comportamiento de las distintas estrategias de actualización implementadas.

  - ***O.O.8:*** *Realizar experimentos y evaluar los resultados.* Se realizarán distintos experimentos de cada una de las estrategias implementadas sobre los distintos escenarios definidos. Dichos experimentos permitirán obtener distintas métricas, como la disponibilidad de la subred y la duración del proceso actualización, permitiendo evaluar el comportamiento de las distintas estrategias en estudio y extraer conclusiones.

1.3 Metodología de la investigación

Con el fin de poder cumplir los objetivos definidos en el apartado anterior se ha seguido la siguiente metodología de investigación que se muestra en el diagrama de la figura 1.1 y que consta de 4 fases principales.

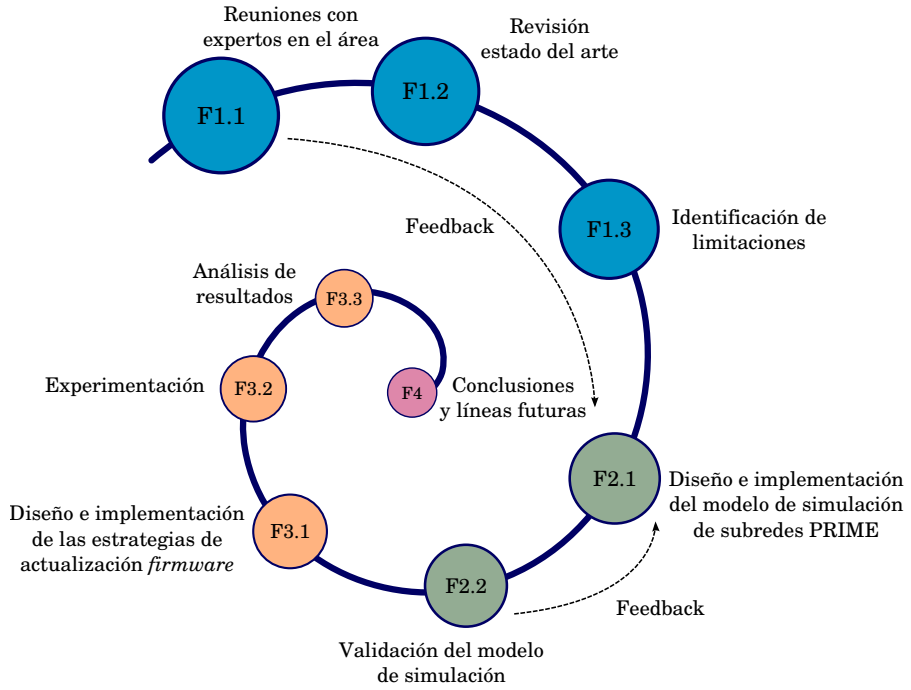


Figura 1.1: Metodología empleada en este trabajo de investigación

La primera fase consiste en la toma de conciencia del problema. En esta primera fase, las reuniones mantenidas con el departamento de telecomunicaciones de Iberdrola han sido de gran ayuda ya que han puesto de relieve sus necesidades e inquietudes como compañía eléctrica y por tanto, expertos en el área. En concreto, Iberdrola muestra su interés en mejorar el proceso de actualización *firmware* de los nodos de servicio de una red PRIME. Para realizar dichas mejoras es necesario contar con un simulador de subredes PRIME capaz de reproducir el comportamiento de redes reales de la forma más fiel posible, ya que las pruebas en campo requieren de mucho tiempo y esfuerzo. Una vez conocido el problema se ha llevado a cabo una exhaustiva revisión del estado del arte en el área de los simuladores de redes PLC de banda estrecha y más concretamente de redes PRIME. Por otro lado, se ha realizado una búsqueda de posibles trabajos que aborden la mejora de la actualización *firmware* de redes PRIME, pero no se han encontrado estudios que se centren en este estándar. Como resultado de esta fase se han identificado las limitaciones de los trabajos

existentes, lo que ha permitido definir la hipótesis y los objetivos de este trabajo de investigación.

La segunda fase, ha consistido en el diseño, desarrollo y validación de un modelo de simulación de subredes PRIME capaz de reproducir el comportamiento de redes reales de este tipo y que permita evaluar la aplicación de actualización *firmware*. La herramienta escogida para el desarrollo del modelo es OMNeT++. El diseño e implementación del modelo de simulación se ha realizado siguiendo la especificación 1.3.6 del estándar PRIME. Una vez implementado, para comprobar que el modelo de simulación funciona correctamente desde el punto de vista funcional, se ha comenzado con la simulación de escenarios más sencillos para después ir incrementando la complejidad de los escenarios hasta haber depurado el código por completo. Además, el modelo ha sido refinado y ajustado gracias al *feedback* recibido, una vez más, por parte del departamento de telecomunicaciones de Iberdrola. A su vez, el modelo de simulación ha sido validado con resultados de pruebas reales aportados por Iberdrola. Para ello, se ha implementado una estrategia de actualización *firmware* en el modelo de simulación que actualmente ya se utiliza en redes reales y se ha comparado el tiempo total necesario para realizar la actualización de los equipos de la red tanto en el entorno real como en el de simulación. Dicha comparación se ha realizado mediante la prueba estadística *t de Student para dos muestras independientes* empleando la herramienta de análisis estadístico R [R C13]. Finalmente, el modelo de simulación desarrollado ha sido publicado en la conferencia *IEEE International Symposium on Power Line Communications and its Applications (ISPLC)* 2015 celebrado en Austin, Texas (EE.UU.).

En la tercera fase, se han diseñado, implementado y analizado distintas estrategias de actualización *firmware* mediante el modelo de simulación de subredes PRIME desarrollado en la fase anterior. Para ello, en primer lugar se han identificado los aspectos de libre implementación de la aplicación de actualización *firmware* definida en el estándar. Después, se han escogido aquellos aspectos que pudieran tener influencia en los resultados de disponibilidad y tiempo de actualización total de red, y se han diseñado e implementado distintas estrategias de actualización realizando variaciones de dichos aspectos. Las estrategias implementadas se han simulado sobre escenarios con distinta topología física y lógica, obteniendo distintas métricas, lo que ha permitido evaluar el comportamiento de las estrategias implementadas.

Por último, en la última fase, se ha llevado a cabo el análisis de los resultados de las simulaciones realizadas. Para realizar dicho análisis, se han procesado los ficheros que contienen los resultados, se han calculado las medias estadísticas y generado los gráficos que recogen dichos resultados mediante la programación de distintos *scripts* de R. Esto ha permitido observar las diferencias entre las distintas estrategias y obtener conclusiones del trabajo realizado. Finalmente, se han definido algunas posibles líneas de investigación futuras.

1.4 Organización de la tesis

En el capítulo 2 se incluye una breve introducción a los sistemas PLC y una descripción de las características más importantes del estándar PRIME que servirá para conocer los fundamentos que se utilizarán a lo largo de este trabajo.

El capítulo 3 incluye una revisión bibliográfica de los distintos simuladores de redes PLC de banda estrecha que aparecen en distintos trabajos de la literatura. La revisión bibliográfica incluida en este capítulo ha permitido realizar un análisis en profundidad de dichas herramientas e identificar los puntos fuertes y débiles de cada solución descrita, lo que ha permitido identificar la necesidad de implementar un modelo de simulación de subredes PRIME propio.

En el capítulo 4 se describe el modelo de simulación de subredes PRIME desarrollado para la realización de este trabajo de investigación. Teniendo en cuenta la importancia de contar con una herramienta que reproduzca fielmente el comportamiento de este tipo de redes, en este capítulo también se describe el proceso de validación del modelo de simulación llevado a cabo.

En el capítulo 5 se describen las distintas estrategias de actualización *firmware* que han sido diseñadas, implementadas y evaluadas mediante el modelo de simulación de subredes PRIME desarrollado. Los resultados de las evaluaciones realizadas también han sido recogidos en este capítulo y permiten conocer cuál es el alcance de la hipótesis planteada.

Por último, el capítulo 6 recoge las conclusiones extraídas tras la realización de este trabajo de investigación y presenta las posibles líneas de investigación futuras que permitan continuar y mejorar el trabajo aquí expuesto.

Capítulo

2

El estándar PRIME

En este capítulo, en la sección 2.1 se incluye una breve introducción a los sistemas PLC, mientras que en la sección 2.2 se describen las partes más importantes de la versión 1.3.6 del estándar PRIME en el que se ha basado el trabajo de investigación realizado. Este capítulo se completa con el apéndice A en el que se recogen algunos aspectos adicionales del estándar.

2.1 Introducción a los sistemas PLC

2.1.1 Breve reseña histórica de PLC

Las primeras aplicaciones de PLC que se pusieron en marcha por las compañías eléctricas incluían comunicaciones de voz y datos a través de las líneas de alta tensión (AT) que normalmente llevan tensiones superiores a 100 kVs y abarcan grandes distancias geográficas [GSW11]. Las líneas de alta tensión han sido utilizadas como un medio de comunicación para voz desde la década de 1920 [Sch09]. En aquellos años la cobertura telefónica era muy escasa y los ingenieros que trabajaban en las centrales eléctricas y centros de transformación utilizaban PLC como una alternativa de comunicación. De este modo, les era posible gestionar las operaciones con compañeros de trabajo que se encontraban a decenas o cientos de kilómetros de distancia.

Más tarde, con la introducción de técnicas de comunicaciones digitales, se comenzaron a alcanzar tasas de datos de unos pocos cientos de bps que servían para llevar a cabo las tareas de medición y telecontrol [Dos01, FLNS10]. Los primeros trabajos en este sentido, de medición y control de carga en las redes eléctricas, fueron llevados a cabo por ingenieros suizos. Después, durante

la Segunda Guerra Mundial, algunos radio aficionados comenzaron a experimentar con PLC ya que sus actividades en el espectro de radio frecuencia (RF) estaban restringidas. En junio de 1954 la *American Institute of Electrical Engineers (AIEE)* publicó el informe que lleva por título *"Guide to Application and Treatment of Channels for Power Line Carrier"*[C$^+$80] que, como el propio título indica, se trata de una guía en el que se incluye un estudio acerca de las características de las líneas eléctricas como canal de comunicación. Este documento destacaba que la tecnología PLC se había vuelto indispensable para la operación de sistemas de energía eléctrica de gran tamaño, por lo que desde entonces se han realizado numerosos trabajos de investigación, especialmente durante los años 80 y 90.

La mayor parte de la investigación en PLC se ha centrado en las redes de distribución de energía eléctrica de baja tensión, ya que geográficamente son las más extendidas y normalmente son las que mayor accesibilidad tienen dentro de los edificios y estructuras. Sin embargo, es importante mencionar que las redes de baja tensión se caracterizan por ser canales de comunicaciones poco usuales y hostiles, ya que cuando fueron diseñadas no se tomaron en cuenta aspectos de las comunicaciones.

Tradicionalmente, las organizaciones o empresas de servicios públicos han jugado un papel de suma importancia en el desarrollo de la tecnología PLC. La principal motivación era la gestión de la carga. La gestión de la carga, también conocida como la gestión de la demanda, es el proceso de equilibrar el suministro de electricidad en la red con la carga o demanda eléctrica. Normalmente, esto se consigue mediante la desconexión selectiva de aquellos aparatos eléctricos que tienen un alto consumo (p.ej. calentador de agua, lavadora, etc.) en los momentos de mayor demanda o *peak demand*. En algunos países, para este mismo propósito se utiliza el sistema "*ripple control*" en el que se emplean relés sensibles a señales de frecuencias determinadas que disparan los interruptores automáticos o disyuntores. El sistema "*ripple control*" es un sistema unidireccional que trabaja a una baja tasa de datos y que opera en la banda de frecuencias menor a 3 kHz. La principal desventaja de este sistema es que para la transmisión de la información puede requerir varios megavatios de potencia. En cambio, los sistemas PLC son muchos más sofisticados y requieren bastante menos potencia para la transmisión de la información y por tanto para llevar a cabo la gestión de la carga [FLNS10].

Otra motivación importante en el desarrollo de sistemas PLC de baja tensión es la medición remota de los contadores. Los desarrollos en este sentido comenzaron en Estados Unidos, donde los salarios de los operarios que leían los contadores eran bastante altos y a las compañías eléctricas no se les permitía cobrar los costes fijos mensuales a sus clientes tal y como ocurría en Europa [HC90]. Además, el tiempo necesario para leer todos los contadores de una red empleando la forma tradicional era muy elevado, hasta el punto en el que en

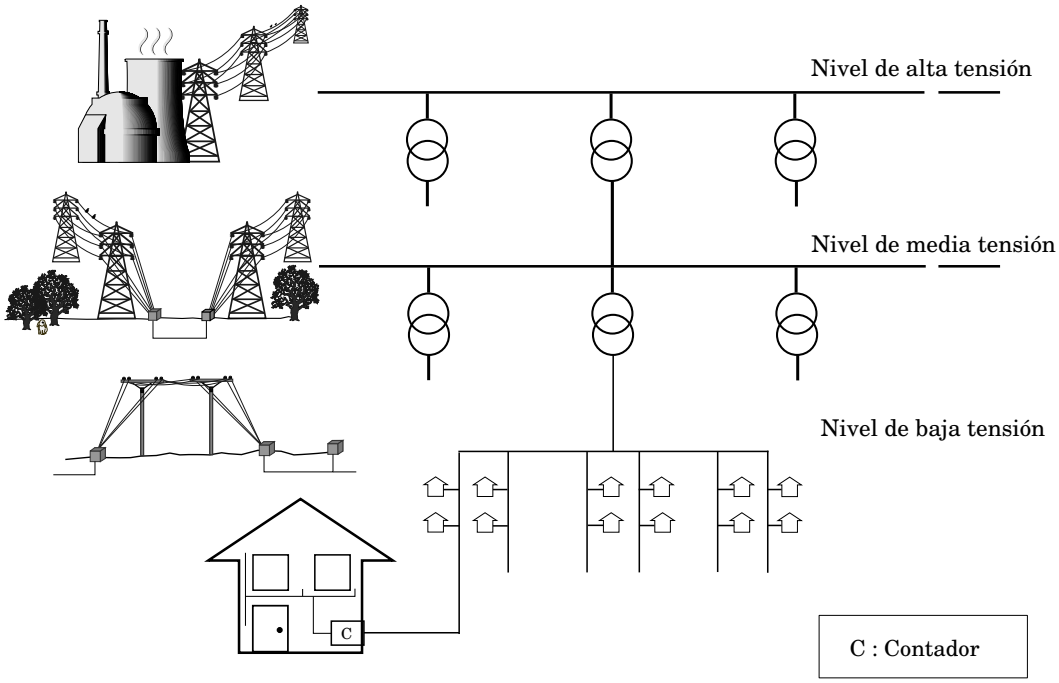


Figura 2.1: Estructura de las redes de suministro eléctricas

el estudio incluido en [Eyr90] se demuestra que una persona encargada de leer contadores de este tipo tiene una tasa media de lectura de tan solo 1 bps. Como se puede observar, se trata de una tasa de lectura muy baja si la comparamos con las posibilidades que nos ofrecen los sistemas modernos PLC.

Más tarde, cuando la tecnología PLC maduró y el espacio de aplicaciones se amplió, surgieron los sistemas basados en *broadband* PLC o PLC de banda ancha que operan en la banda de frecuencias alta (de 1,8 MHz a 250 MHz) y alcanzan velocidades de transmisión de varios cientos de Mbps. Por lo tanto, actualmente, los sistemas PLC se dividen en dos grupos principales: PLC de banda estrecha (trabaja en la banda inferior a 500 kHz) que permite servicios de comunicaciones con tasas de datos relativamente bajas (hasta 200 kbps) y garantiza la realización de varias aplicaciones de automatización y control, así como un par de canales de voz, y los sistemas PLC de banda ancha que permiten velocidades de datos más allá de los 2 Mbps y permiten la realización de los típicos servicios de telecomunicaciones en paralelo, como la telefonía y el acceso a Internet [HHL05].

2.1.2 Redes de acceso PLC

Los sistemas de suministro eléctrico, como se muestra en la figura 2.1, se componen de tres niveles de red y se detallan a continuación:

- **Alta tensión ($> 100\ kV$).** Las redes de alta tensión conectan las centrales eléctricas con grandes redes de suministro o grandes clientes. Por lo general, abarcan distancias muy grandes lo que permite el intercambio de energía dentro de un continente. Las redes de alta tensión, normalmente, se componen de cables o líneas de suministro aéreas.

- **Media tensión (** $10-100\ kV$ **).** Las redes de media tensión dan suministro a áreas grandes como ciudades, o a clientes industriales o comerciales de gran tamaño. Las distancias que abarcan son significativamente más pequeñas que las redes de alta tensión. Por otro lado, cabe destacar que las redes de media tensión están formadas tanto por cables aéreos como por subterráneos.

- **Baja tensión (** $230/400\ V,\ 110\ V\ en\ EE.UU.$ **).** Las redes de baja tensión suministran energía a los usuarios finales ya sean clientes individuales o usuarios individuales de un cliente más grande. Su longitud suele llegar hasta unos pocos cientos de metros. Por otro lado, en áreas urbanas las redes de baja tensión están compuestas por cables subterráneos, mientras que en áreas rurales normalmente se pueden encontrar como redes aéreas.

Las redes de suministro de baja tensión se componen de un transformador de potencia y un número de líneas de suministro de energía que conectan los usuarios finales a través de contadores de energía eléctrica. Estas redes de baja tensión están disponibles en un gran número de hogares de todo el mundo, por lo que pueden ser utilizadas para proporcionar soluciones de última milla o "*last mile*" a través de redes de acceso PLC. El término de última milla se comenzó a utilizar en telefonía para referirse a la conexión entre el abonado y la central telefónica. El conjunto de conexiones entre los abonados y las centrales forman la llamada red de acceso. Estos términos también pueden aplicarse a la tecnología PLC.

Como se observa en la figura 2.1, las redes de suministro de baja tensión están conectadas a las redes de media y alta tensión a través de un transformador. En cambio, cuando hablamos de las redes de acceso PLC, éstas se conectan a la red troncal de comunicaciones o *Wide Area Network (WAN)* a través de un nodo base o *Base Node (BN)* que normalmente se coloca junto al transformador [Mak01, GYC+08]. La redes troncales de comunicaciones son las encargadas de agregar grandes volúmenes de datos y transportarlos entre puntos muy distantes. Pueden estar formadas por todo tipo de medios de comunicación y dispositivos que conectan distintas subestaciones de generación, de transporte e incluso de distribución. Para establecer estas conexiones, generalmente se utilizan cables de fibra óptica y enlaces microondas. En la capa de transporte, se emplean tecnologías como *Plesiochronous Digital Hierarchy (PDH)*, *Synchronous Digital Hierarchy (SDH)* y *Coarse Wavelength Division Multiplex (CWDM)* o *Dense Wavelength Division Multiplex (DWDM)*. Las velocidades de transmisión de

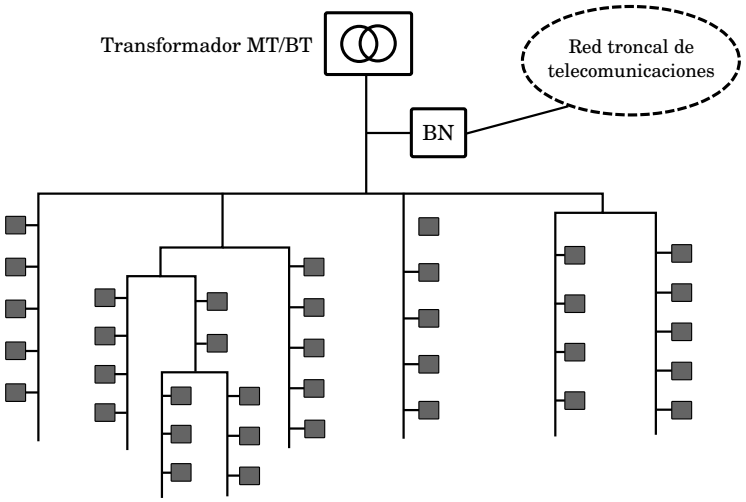


Figura 2.2: Estructura de la red de acceso PLC en redes de baja tensión

estos enlaces van desde 2 Mbps a 2,5 o 10 Gbps (En el caso de STM-16 o STM-64 respectivamente, cuando hablamos de SDH) [FLNS10]. Muchas empresas de servicios que suministran energía eléctrica tienen sus propias redes de telecomunicaciones que conectan con sus transformadores y pueden ser utilizadas como una red troncal. En cambio, si este no es el caso, los transformadores también pueden conectarse a una red de telecomunicaciones convencional [HHL05]. En la figura 2.2 se muestra un ejemplo de la estructura de una red de acceso PLC de la red de baja tensión.

La señal de comunicaciones que llega desde la red troncal tiene que ser convertida a una forma en la que sea posible su transmisión a través de la red de baja tensión. Esta conversión se lleva a cabo en el nodo base del sistema PLC. Por otro lado, en el otro de la red de acceso, los suscriptores se conectan a la red a través de un módem PLC colocado en el contador eléctrico o conectado a cualquier toma de la red eléctrica interna de la vivienda, lo que se conoce como la solución *in-home* PLC. El módem convierte la señal recibida de la red PLC en una forma estándar que puede ser procesada por los sistemas de comunicaciones convencionales.

Por lo tanto, los elementos más importantes de una red de acceso PLC son:

- Módem PLC (M): conectado al contador de energía eléctrica.
- Nodo Base (BN): junto al transformador.

Estos elementos de red se aseguran de que la conversión de la señal se haga adecuadamente y que la información sea transmitida por las líneas eléctricas. Del mismo modo, estos mismos elementos también se encargarán de recibir y transformar correctamente la señal recibida.

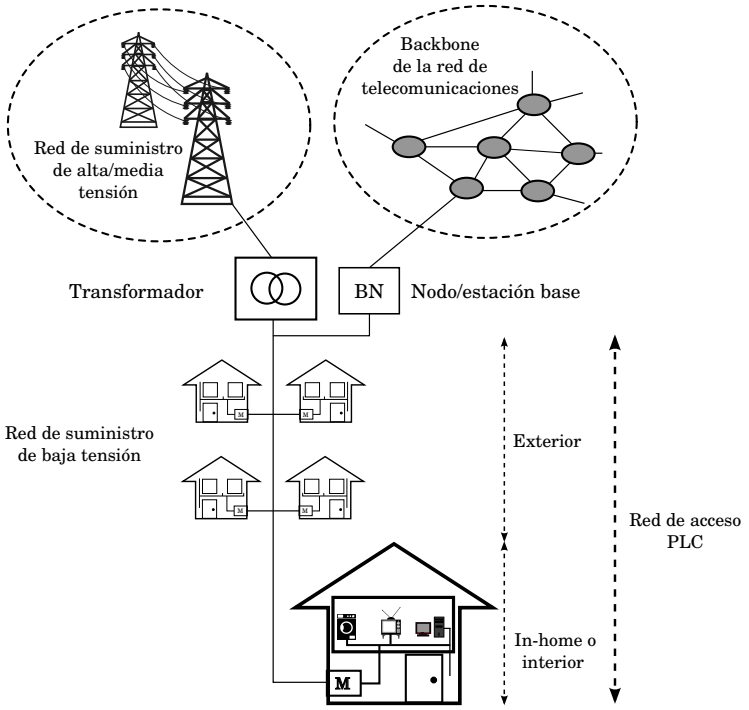


Figura 2.3: Estructura de la red de acceso PLC

2.1.3 Estandarización de sistemas PLC

Tal y como se ha adelantado al final del apartado 2.1.1 de este capítulo, se distinguen dos tipos de sistemas PLC: los sistemas PLC de banda ancha y los de banda estrecha. En las siguientes secciones se describirán algunos de los estándares más importantes de ambos tipos de sistemas.

2.1.3.1 PLC de banda ancha

Las soluciones PLC de banda ancha trabajan en bandas de frecuencia que van desde los 1,8 MHz a los 250 MHz y pueden alcanzar velocidades de hasta unos cientos de Mbps. Están destinados a proporcionar acceso a Internet a los consumidores y a formar parte de distintas aplicaciones de red doméstica. Entre los estándares más importantes se encuentran los siguientes: TIA-1113, IEEE 1901 (FFT-OFDM y Wavelet-OFDM) y ITU-T G.hn [FLNS10].

- **TIA-1113.** Se trata del primer estándar de PLC de banda ancha aprobado por *American National Standards Institute (ANSI)*. El estándar se basa en gran medida en las especificaciones HomePlug 1.0 [All01] que cuenta con una velocidad de comunicaciones de hasta 14 Mbps y emplea la técnica de transmisión *Orthogonal Frequency Division Multiplexing (OFDM)*, o como se conoce en castellano, multiplexación por división de frecuen-

cias ortogonales [FLNS10]. La idea básica de OFDM es dividir el espectro disponible en varias subportadoras de banda estrecha y baja velocidad de datos. Para obtener una alta eficiencia espectral, las respuestas en frecuencia de las subportadoras se superponen y son ortogonales entre sí (desfase de 90º entre señales de la misma frecuencia) tal y como indican las siglas OFDM. Cada subportadora puede ser modulada empleando distintos esquemas de modulación como *Binary Phase Shift Keying (BPSK)* o *Quaternary Phase Shift Keying (QPSK)*, que se escogerá en función de la calidad del canal o de la funcionalidad operativa. Por otro lado, el mecanismo *Forward Error Correction (FEC)* o de corrección de errores utilizado se basa en códigos *Reed Solomon (RS)* o *Convolutional Code (CC)*.

- **IEEE 1901.** El grupo de trabajo IEEE 1901 se estableció en 2005 para unificar tecnologías *power line* con el objetivo de desarrollar un estándar para dispositivos de comunicación de alta velocidad ($> 100 Mbps$) que utilizan frecuencias menores a los 100 MHz y que abordan tanto aplicaciones para redes domésticas o *Home Area Network (HAN)* como de acceso [FLNS10, GL08, GT10, RHL+11]. El estándar fue aprobado en octubre de 2008 y define dos capas físicas, una basada en FFT-OFDM [AKYN05] y otra basada en Wavelet-OFDM [GKK08]. La elección de qué capa física implementar es libre, por lo que cada fabricante puede optar por implementar ambos tipos o no.

  La opción de capa PHY que implementa FFT-OFDM incluye el mecanismo FEC basado en Turbo Code. La opción basada en Wavelet-OFDM, en cambio, utiliza un esquema FEC que concatena códigos RS y CC. Opcionalmente, en el caso de Wavelet-OFDM también se puede emplear la codificación basada en *Low Density Parity Check (LDPC)*. Por otro lado, los esquemas de modulación soportados se enumeran a continuación: QAM, QPSK, BPSK y modulación ROBO [G+10].

  Sobre estas dos capas PHY se definen dos capas MAC: una para aplicaciones de redes domésticas y otra para proporcionar acceso a Internet. Es importante señalar la necesidad de la existencia de dos capas MAC ya que cada aplicación tiene distintos requisitos.

  Para manejar la coexistencia entre distintas capas PHY y MAC se desarrolló el protocolo *Inter-System Protocol (ISP)*. Este protocolo permite que distintos dispositivos PLC basados en IEEE 1901 puedan compartir el medio a pesar de sus diferencias de la capa PHY. Además, el ISP también permite que dispositivos IEEE 1901 puedan convivir con dispositivos basados en el estándar ITU-T G.hn.

- **ITU-T G.hn.** En el año 2006, la ITU-T comenzó a trabajar en el proyecto G.hn con el objetivo de desarrollar una recomendación a nivel mun-

Tabla 2.1: Resumen de características de los estándares de PLC de banda ancha

	TIA-1113	**IEEE 1901**	**ITU-G.hn**
Capas	PHY, MAC	PHY, MAC	PHY, DL
Multiplexación	OFDM	FFT-OFDM Wavelet-OFDM	FFT-OFDM
Velocidades	14 Mbps	>100 Mbps (hasta 500 Mbps)	<1 Gbps
Banda de frecuencias	4,5 MHz - 21 MHz	<100 MHz	<100 MHz
Modulación	BPSK QPSK	QAM QPSK BPSK ROBO	QAM
FEC	RS	Turbo Code (FFT-OFDM) RS, CC y LDPC (Wavelet-OFDM)	LDPC
Otros	-	ISP	ISP opcional

dial para desarrollar un transceptor HAN unificado capaz de operar sobre cualquier tipo de cableado que pueda haber en el hogar: líneas telefónicas, cableado eléctrico, cables coaxiales y Cat-5 [Tel01] con velocidades de hasta 1 Gbps. La capa PHY de G.hn fue ratificada por la ITU-T en octubre de 2009 como la recomendación G.9960 [G.909], mientras que la capa de enlace de datos se ratificó en junio de 2010 como la recomendación G.9961 [G.910]. Este estándar está destinado a emplearse en casas residenciales y lugares públicos como en oficinas pequeñas, viviendas u hoteles, pero al contrario que el estándar IEEE 1901, no aborda las aplicaciones de acceso PLC. Además, los transceptores que cumplan las recomendaciones G.9960/G.9961 no requieren que tengan que convivir con otras tecnologías, por lo que no necesariamente tienen que soportar el protocolo ISP.

Este estándar define una sola capa física basada en FFT-OFDM y cada una de las subportadoras se modula siguiendo el esquema *Quadrature Amplitude Modulation (QAM)*. Por otro lado, el mecanismo de codificación empleado es el LDPC.

La tabla 2.1 incluye un resumen de características de los estándares PLC de banda ancha que se acaban de describir.

Tabla 2.2: Bandas de frecuencia CENELEC para PLC

Banda	Frecuencia (kHz)	Uso
A	9 - 95	Compañías eléctricas
B	95 - 125	Cualquier aplicación
C	125 - 140	Sistemas de redes en el hogar
D	140 - 148,5	Sistemas de alarmas y seguridad

2.1.3.2 PLC de banda estrecha

Como se ha visto en el apartado anterior 2.1.3.1, los sistemas PLC de banda ancha están destinados a proporcionar acceso a Internet a los usuarios así como a formar parte de aplicaciones de red doméstica. Los sistemas de banda ancha maximizan la tasa de datos a costa de perder alcance en las comunicaciones, pero debido a que para las compañías eléctricas la cobertura de las comunicaciones es más importante que la velocidad de datos, éstas emplean PLC de banda estrecha para sus aplicaciones, ya que de este modo pueden llegar a zonas más lejanas y por tanto pueden comunicarse con un mayor número de usuarios [DBS11].

Los sistemas PLC de banda estrecha se definen como aquellos que operan en las bandas de frecuencia inferiores a 500 KHz. Generalmente, este tipo de sistemas trabajan en el rango de frecuencias especificado por la norma CENELEC. El estándar CENELEC EN 50065 [CEN93] permite la comunicación a través de las líneas de baja tensión (BT) en el rango de frecuencias de 3 kHz a 148,5 kHz. En ella se definen cuatro bandas de frecuencia que se muestran en la tabla 2.2. Como se puede observar, el uso de la banda A está limitado a las compañías eléctricas ya que está reservada para aplicaciones que tienen que ver con la gestión y control de las redes.

Por otro lado, los estándares PLC de banda estrecha más conocidos en la actualidad se describen a continuación:

- **PowerLine Intelligent Metering Evolution (PRIME)**[pri06]: fue creado en 2006 por la denominada Alianza PRIME (liderada por Iberdrola, Atmel, ZIV, etc.) para proporcionar un estándar abierto y público para soluciones PLC de banda estrecha. Fue diseñado para aplicaciones de contadores inteligentes y otras funcionalidades de *Smart Grid*s. Opera en la banda de CENELEC-A (de 41,992 kHz a 88,867 kHz) empleando la técnica de multiplexación OFDM con la señal cargada en 97 subportadoras (96 de datos y una piloto). Por otra parte, el estándar permite escoger entre las siguientes modulaciones: *Differential Binary Phased Shift Keying (DBPSK)*, *Differential Quadrature Phased Shift Keying (DQPSK)* y *Differential 8 Phased Shift Keying (D8PSK)*. Por otro lado, el mecanismo de corrección de errores em-

pleado se basa en un código convolucional con una tasa de 1/2. En función de la modulación empleada y de si se emplea el código convolucional o no, la velocidad de transmisión que se puede llegar a alcanzar es de unos 130 kbps.

PRIME define las capas física (PHY) y de acceso al medio (MAC), y sirve a través de las capas de convergencia IEC 61334-4-32 e IPv6 a aplicaciones como *Device Language Message Specification/Companion Specification for Energy Metering (DLMS/COSEM)* que proporcionan un protocolo para realizar la lectura de contadores. Una de las principales características de PRIME es la autoconfiguración de la red. PRIME define mecanismos en la capa MAC que permiten que la red pueda reconfigurarse dinámicamente en cada momento de manera que los nodos necesarios mantienen el contacto con la red.

- **G3-PLC** [g3p09]: se trata de una especificación abierta desarrollada por la Alianza G3 promovida por EDF y MAXIM, que incluye las capas PHY, MAC y el perfil de medida. Opera en la banda de CENELEC-A (de 35,938 kHz a 90,625 kHz) y emplea la técnica de multiplexación OFDM con la señal cargada en 36 subportadoras. Las modulaciones empleadas por G3 son DBPSK, DQPSK y D8PSK. Por otro lado, el sistema puede trabajar en dos modos distintos: en modo normal y en modo robusto. En el modo normal, el mecanismo FEC se compone de un codificador *Reed Solomon* y uno convolucional. En el modo robusto, en cambio, el FEC se compone de los codificadores *Reed Solomon* y convolucional seguido de un *Repetition Code* o código de repetición. En función del modo escogido la velocidad de transmisión será distinta, pudiendo alcanzar hasta un máximo de 33,4 kbps. Por otro lado, G3 también es capaz de trabajar en una combinación de dos o más bandas de CENELEC, como en BC, BCD y BD [DBDAT14], para soportar otro tipo de aplicaciones, como las que se incluyen en la tabla 2.2.

- **METERS AND MORE (M&M)** [met10]: se trata del protocolo de comunicaciones desarrollado por la asociación del mismo nombre, que permite una transferencia de datos bidireccional en un sistema *Advanced Metering Infrastructure (AMI)*. Se ha construido sobre el protocolo Telegestore de ENEL y es el empleado por ENDESA [end44] en sus despliegues. M&M cumple con los requisitos de funcionalidad, seguridad y comunicaciones definidos en el proyecto OPEN Meter [ope11]. Emplea modulación BPSK junto con la técnica de corrección de errores basada en un código convolucional 1/2 y alcanza velocidades de hasta 4,8 kbps. Para la transmisión de datos emplea una subportadora centrada en 86 kHz (en la banda CENELEC A). Por último, M&M cubre el *stack* completo del protocolo desde la capa física a la de aplicación.

- **Embedded Communications Software Stack (ECSS)**[SPMGN12]: se trata de un estándar creado por ADD Semiconductor (ahora Atmel) y ha sido utilizado para distintas aplicaciones como el *Smart Metering* o medición inteligente, *Smart Lighting* o alumbrado inteligente y *Home Automation* o automatización del hogar. En función de la aplicación por tanto, ECSS trabaja en una banda CENELEC distinta. Por otra parte, utiliza la modulación *Frequency Shift Keying (FSK)* o modulación por desplazamiento de frecuencia, con la opción de emplear un código convolucional como mecanismo de corrección de errores y puede alcanzar una velocidad de transmisión de hasta 4,8 kbps.

La tabla 2.3 incluye un resumen de características de los estándares PLC de banda estrecha que se acaban de describir.

Tabla 2.3: Resumen de características de los estándares de PLC de banda estrecha

	PRIME	**G3**	**ECSS**	**M&M**
Creador	PRIME Alliance	G3 Alliance	ADD Semiconductor	Asociación M&M
Capas	PHY MAC	PHY MAC	PHY MAC	PHY MAC LLC APP
Multiplexación	OFDM	OFDM	-	-
Modulación	DBPSK DQPSK D8PSK	DBPSK DQPSK D8PSK	FSK	BPSK
FEC	CC	RS,CC, RC	CC	CC
Banda de frecuencias	41-89 KHz (CENELEC-A)	35 - 91 KHz (CENELEC-A) y resto de bandas CENELEC	60 - 132,5 KHz (CENELEC-A,B y C)	86 KHz (CENELEC-A)
Número de portadoras	97	36	1	1
Velocidad	128,6 Kbps	33,4 Kbps	4,8 Kbps	4,8 Kbps

2.2 Estándar PRIME

Después de presentar los distintos estándares de PLC de banda estrecha existentes en la actualidad, en esta sección se incluye una descripción más detallada del estándar en la que se centra este trabajo de investigación, es decir, la versión 1.3.6 de PRIME.

2.2.1 Visión general

El estándar PRIME define las capas más bajas de un sistema de transmisión de datos de banda estrecha PLC a través de la red eléctrica. La figura 2.4 muestra las capas de comunicación y el alcance de la especificación.

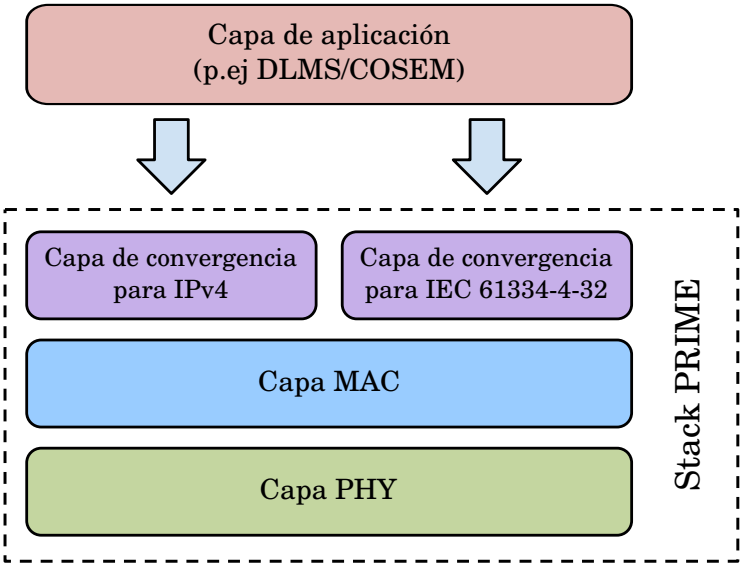


Figura 2.4: Pila o stack del protocolo PRIME

Como se observa en dicha figura, el *stack* de PRIME se compone de tres capas principales, que son la capa de convergencia (CL), la capa (MAC) y la capa física (PHY).

La capa CL clasifica el tráfico asociándolo con su conexión MAC correcta. Esta capa realiza el mapeo de cualquier tipo de tráfico de modo que se incluya correctamente en tramas *MAC Service Data Unit (MSDU)*. La capa de convergencia está dividida en dos subcapas: una subcapa común conocida como *Common Part Convergence Layer (CPCS)* y varias subcapas especificas llamadas *Service Specific Convergence Sublayer (SSCS)* para adaptar distintos tipos de tráfico a los distintos MSDUs.

La capa MAC proporciona funcionalidades básicas MAC de acceso al sistema, de asignación de ancho de banda, gestión de la conexión y resolución de la topología.

La capa PHY transmite y recibe tramas de tipo *MAC Protocol Data Unit (MPDU)* dentro de tramas *PHY Protocol Data Unit (PPDU)* entre nodos vecinos. Se basa en la técnica de multiplexación OFDM que opera en la banda CENELEC-A alcanzando una tasa de transmisión de hasta 130 kbps aproximadamente.

La especificación PRIME toma los puntos fuertes de las tecnologías más avanzadas, adaptándolas a sus necesidades, simplificando los procesos, cabeceras,

etc, para asegurar la interoperabilidad entre los dispositivos y distintas implementaciones de los elementos del sistema.

2.2.2 Capa PHY

La capa PHY de PRIME está diseñada para transmitir y recibir datos a través de líneas eléctricas que fueron originalmente diseñadas para la distribución de energía eléctrica a 50-60 Hz CA, en lugar de para las comunicaciones [FLNS10]. El uso de este medio para las comunicaciones presenta algunos problemas técnicamente desafiantes tal y como se verá a continuación.

- **Propagación multicamino**: es el fenómeno que ocurre cuando la señal enviada por el transmisor llega al receptor por dos o más caminos y en diferentes tiempos. Este fenómeno puede causar problemas en la recepción de la señal, debido a que se reciben múltiples versiones de la señal transmitida desplazadas una respecto de otra en términos de tiempo y fase. Las fases y amplitudes de las diferentes componentes causan fluctuaciones en la intensidad de la señal, introduciendo desvanecimientos, distorsión o ambas a la vez.

  En los sistemas PLC este fenómeno puede ser debido a que los cables que componen las redes eléctricas están divididos de forma asimétrica y cuentan con numerosas conexiones irregulares entre las distintas secciones de la red y los clientes, así como entre transiciones entre cables aéreos y subterráneos (figura 2.1). Estas transiciones en los cables pueden causar reflexiones y cambios en las impedancias características [ZD00]. Como resultado, se obtiene la propagación multicamino de la señal con desvanecimiento selectivo en frecuencia. La principal característica del desvanecimiento selectivo en frecuencia es que algunas de las componentes frecuenciales de la señal transmitida se amplifican mientras que otras se atenúan [PHB11].

- **Atenuación**: hace referencia a la pérdida de potencia sufrida por la señal al ser transmitida por cualquier medio de transmisión.

  La atenuación en el canal PLC depende de la frecuencia por lo que, como ya se ha adelantado, algunas frecuencias se verán más atenuadas que otras. Dicha atenuación también dependerá de la distancia entre el emisor y receptor ya que, por lo general a medida que la distancia aumenta, la potencia de la señal transmitida se verá disminuida. El trabajo incluido en [Dos01] demuestra que la atenuación en cables relativamente cortos (cables hasta 200-300 m) es aceptable, mientras que en cables más largos para poder establecer la comunicación entre pares de nodos es necesario recurrir a la técnica de repetición.

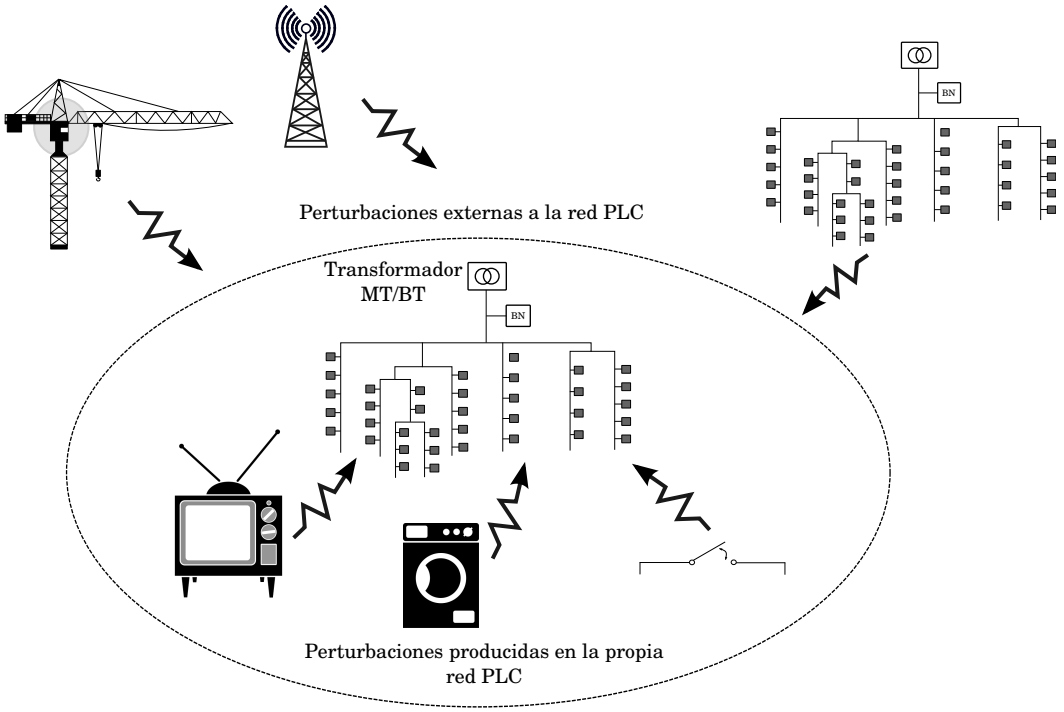


Figura 2.5: Influencia de algunos tipos de fuentes de perturbaciones

- **Ruido**: se considera ruido a todas las perturbaciones eléctricas que interfieren sobre las señales transmitidas o procesadas.

  A diferencia de lo que ocurre en los sistemas de comunicación convencionales, el ruido presente en sistemas PLC es intenso, no-blanco, variable en el tiempo y a menudo no Gaussiano. Este ruido es la suma de los distintos ruidos producidos y emitidos tanto por los elementos de la propia red PLC como por los de fuera de ella. La figura 2.5 muestra distintos tipos fuentes de ruido tanto internas como externas a la propia red.

  En la literatura se proponen diversas formas de clasificar los distintos tipos de ruido presentes en los sistemas PLC [ZD02a, GRD04, FLNS10, HS00, HHL05, CDCL09] como, por ejemplo, la que se incluye a continuación:

  – *Ruido de fondo de color.* Este tipo de ruido se caracteriza por una densidad espectral de potencia relativamente baja que aumenta significativamente en las frecuencias más bajas, ya que la propagación del ruido entre cada fuente y receptor sufre una mayor atenuación a frecuencias más altas. Al contrario de lo que ocurre con el ruido blanco, que es un ruido aleatorio y tiene una densidad espectral uniforme, el ruido de fondo de color tiene una fuerte dependencia de la frecuencia. El ruido de fondo puede ser causado por el uso de elec-

trodomésticos convencionales como ordenadores, televisores, secadores de pelo, etc. En este ruido de fondo también se incluye el ruido de tipo térmico causado por los amplificadores *front-end* de los propios receptores.

- *Ruido impulsivo*. El ruido impulsivo es aquel ruido cuya intensidad aumenta bruscamente durante un impulso. En los sistemas PLC a menudo aparece este tipo de ruido que cuenta con grandes amplitudes y duraciones cortas que van desde unos pocos micro-segundos a mili-segundos. El ruido impulsivo puede ser clasificado en tres tipos principales:
  - *Ruido impulsivo cíclico síncrono a la red eléctrica*. La forma de onda de este tipo de ruido se compone de un tren de impulsos con una frecuencia igual o doble a la de la red eléctrica. Una causa típica de este tipo de ruido son los reguladores de luz basados en rectificadores controlados de silicio o en tiristores.
  - *Ruido impulsivo cíclico asíncrono a la red eléctrica*. La forma de onda de este tipo de ruido se compone de un tren de impulsos con una frecuencia mucho mayor que la frecuencia de la red eléctrica. Una causa típica de este tipo de ruido es el uso de fuentes de alimentación conmutadas que se encuentran en diferentes electrodomésticos de hoy en día.
  - *Ruido impulsivo aislado*. El ruido impulsivo aislado se compone de impulsos que ocurren en intervalos de tiempo aleatorios. Este tipo de ruido ocurre, por ejemplo, cuando un interruptor de pared o el termostato de un calentador corta o restablece la alimentación eléctrica.
- *Ruido de banda estrecha.* Los sistemas PLC de banda ancha comparten bandas de frecuencia con sistemas de comunicación de difusión e inalámbricos, y las señales radio de estos sistemas pueden contaminar los canales PLC a través de ruido de banda estrecha. Este tipo de ruido se conoce como *tone jammer*.

Cuando un receptor PLC se encuentra cerca de una fuente de ruido y no hay ningún otro dispositivo cerca del receptor, dicha fuente de ruido dominará la forma de onda del ruido recibido. En este caso el ruido puede ser clasificado en una de las clases de ruido mencionadas. Sin embargo, en un entorno típico, donde hay muchos dispositivos conectados a la línea, el ruido presente en el sistema PLC es la superposición de distintas formas de onda de distintas clases.

A continuación, se detallarán algunas de las características mas importantes de la capa PHY del estándar PRIME entre las que se encuentran: la técnica

de transmisión, banda de frecuencias, modulaciones, técnicas de codificación empleadas y el formato de la trama PHY.

2.2.2.1 Técnica de transmisión

Una de las novedades de PRIME es el uso de *Orthogonal Frequency Division Multiplexing (OFDM)* en lugar de emplear soluciones de portadora simple tradicionales que se han utilizado en el pasado para las comunicaciones PLC de banda estrecha [PRI08].

La principal ventaja de OFDM sobre los esquemas de portadora simple es su capacidad de hacer frente a las condiciones adversas de canal, como por ejemplo, la atenuación de las altas frecuencias en las líneas eléctricas de gran longitud, el ruido y el desvanecimiento selectivo en frecuencia debido al fenómeno *multipath* (propagación multicamino), sin la necesidad de añadir complejos mecanismos adicionales (p.ej. filtros de ecualización).

2.2.2.2 Banda de frecuencias

La banda de frecuencias escogida es la CENELEC-A definida en la norma EN50065-1. Esta banda de frecuencias va desde los 3kHz a lo 95 kHz. En concreto, el rango de frecuencias empleado por PRIME va desde los 41 kHz a los 89 kHz aproximadamente. La primera subportadora está centrada en la frecuencia 41,992 kHz, mientras que la última está centrada en la frecuencia 88,867 kHz.

2.2.2.3 Modulación digital

PRIME permite escoger entre tres distintas modulaciones, de modo que permite optimizar la velocidad de datos en cada enlace en función de las condiciones del canal. Las modulaciones posibles a emplear son DBPSK, DQPSK y D8PSK.

2.2.2.4 FEC

Además de la robustez inherente de la propia señal de OFDM sin codificar, es fundamental realizar algún tipo de codificación de canal para los casos en los que el valor de la relación señal a ruido (SNR) sea baja. Para ello, se emplea una codificación convolucional que ha demostrado ser capaz de lidiar con un amplio rango de patrones de ruido en las medidas reales realizadas en distintos escenarios. El uso del mecanismo de codificación es opcional y las capas superiores decidirán de forma adaptativa si conviene utilizarla o no.

2.2.2.5 Formato de la trama PHY

La figura 2.6 muestra la estructura de la trama PHY. Como se puede observar, la trama comienza con un preámbulo de 2,048 ms de duración, seguido por un determinado número de símbolos OFDM, cada uno de los cuales dura 2,24 ms. Los primeros dos símbolos OFDM corresponden a la cabecera de la trama PHY, mientras que los M símbolos OFDM restantes transportan la carga útil o *payload*. El valor de M se incluye en la cabecera PHY y puede tener un valor máximo de hasta 63. En la sección A.1.1 del apéndice A se muestra con más detalle la estructura de la trama PHY.

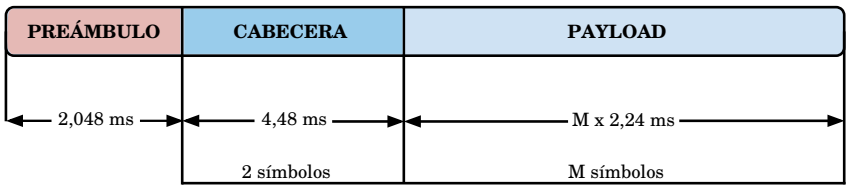


Figura 2.6: Formato de la trama PHY

Como ya se ha adelantado en el apartado 2.2.2.3, PRIME permite escoger entre tres distintos esquemas de modulación, de modo que permite optimizar la velocidad de datos en cada enlace en función de las condiciones del canal. En la tabla 2.4, se muestra la tasa de datos de la capa PHY durante la transmisión del *payload*.

Tabla 2.4: Tasas de transmisión de la capa PHY

	DBPSK		**DQPSK**		**D8PSK**	
Código convolucional (1/2)	On	Off	On	Off	On	Off
Bits de información por subportadora N_{BPSC}	0,5	1	1	2	1,5	3
Bits de información por símbolo OFDM	48	96	96	192	144	288
Tasa de datos en bruto (kbps aprox.)	21,4	42,9	42,9	85,7	64,3	128,6

Por otro lado, la tabla 2.5 muestra los datos del esquema de modulación y codificación empleado a la hora de transmitir la cabecera PHY. Como se puede observar, para la transmisión de la cabecera se emplea la modulación más robusta entre todas las posibilidades.

Tabla 2.5: Esquema de modulación y codificación, y el tamaño de la porción de la cabecera en la trama PHY

	DBPSK
Código convolucional (1/2)	On
Bits de información por subportadora N_{BPSC}	0,5
Bits de información por símbolo OFDM	42

2.2.3 Capa MAC

2.2.3.1 Visión general

El sistema PRIME se compone de subredes. Cada una de estas subredes puede ser vista desde un punto de vista lógico como un árbol y se compone de los siguientes tipos de nodos: el nodo base y los nodos de servicio.

- **Nodo base**. Es la raíz de la estructura de árbol de la red y actúa como nodo maestro que proporciona conectividad a todos los elementos de la subred. A su vez, es el encargado de gestionar los recursos y conexiones de la subred. Hay que destacar que solamente hay un nodo base por cada subred. Al principio, el nodo base es la subred en sí y los demás nodos que quieran formar parte de dicha subred deberán seguir un proceso de registro.

- **Nodo de servicio**. Son las hojas o puntos de ramificación de la estructura de árbol de la red. Inicialmente, los nodos de servicio se encuentran en un estado funcional de *Desconectado* y siguen un proceso de registro para formar parte de la subred. Estos nodos tienen dos funciones principales: mantener la conectividad con la subred para sus capas de aplicación y conmutar los paquetes de datos de otros nodos para propagar la conectividad a través de la subred.

Por otra parte, los nodos de servicio cambian su comportamiento dinámicamente y pueden pasar de actuar como *Terminal* a *Switch* y viceversa, en respuesta a unos eventos predefinidos en la red.

La figura 2.7 muestra el diagrama de transición de estados funcionales de un nodo de servicio. Los tres estados funcionales de un nodo de servicio son los siguientes: *Desconectado*, *Terminal* y *Switch*.

- **Desconectado**. Se trata del estado funcional inicial para todos los nodos de servicio. Cuando un nodo se encuentra en este estado inicial no es capaz de comunicarse ni de conmutar paquetes de otros nodos, por lo que su función principal es la de buscar una subred que esté a su alcance y tratar de registrarse en ella.

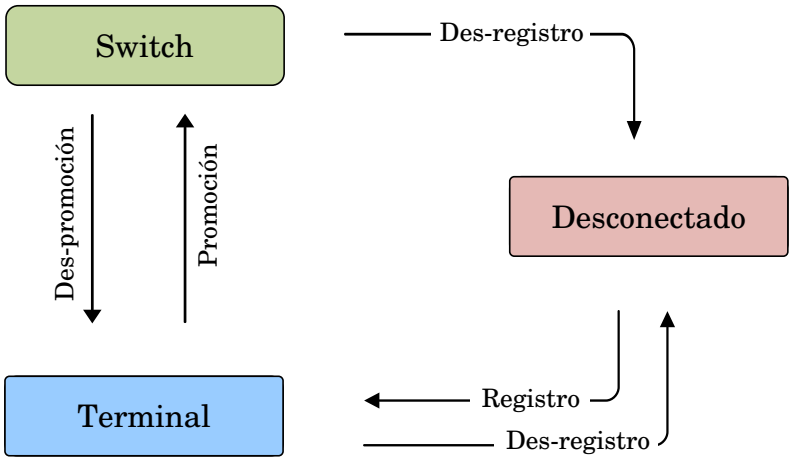


Figura 2.7: Diagrama de transiciones de los nodos de servicio

- **Terminal**. Cuando un nodo de servicio se encuentra en este estado funcional es capaz de establecer conexiones y transmitir datos, pero no es capaz de conmutar los paquetes de otros nodos.

- **Switch**. Cuando el nodo de servicio se comporta como *Switch* es capaz de llevar a cabo todas las funciones que realiza en el estado *Terminal*. Además, también será capaz de enviar datos desde y hacia otros nodos en la misma subred. Por lo tanto, se trata de un punto de ramificación de la estructura de árbol.

Los eventos y procesos asociados que provocan los cambios de un estado funcional a otro son los siguientes:

- **Registro**. Se trata del proceso mediante el cual un nodo de servicio se incluye a sí mismo en la lista de nodos registrados que tiene el nodo base. Cuando el proceso de registro se completa de forma satisfactoria significa que el nodo de servicio pasa a formar parte de la subred. Por lo tanto, representa la transición del estado *Desconectado* a *Terminal*.

- **Des-registro**. Se trata del proceso mediante el cual el nodo de servicio se elimina a sí mismo de la lista de nodos registrados que tiene el nodo base. El proceso de des-registro puede ser iniciado tanto por el nodo de servicio como por el nodo base. Existen varios motivos por los cuales puede darse un proceso de des-registro de un nodo de servicio. Por un lado, un nodo de servicio puede des-registrarse a sí mismo para tratar de encontrar un punto de enganche mejor, es decir, para cambiar de nodo *Switch* mediante el cual el nodo de servicio se ha unido a la subred. Por otro lado, un nodo base puede realizar el des-registro de un nodo de servicio como resultado

de un fallo de cualquier procedimiento MAC. Si el proceso de des-registro se completa correctamente, el nodo de servicio pasaría al estado *Desconectado* y dejaría de formar parte de la subred.

- **Promoción**. Se trata del proceso mediante el cual un nodo de servicio adquiere la capacidad para conmutar (repetir, reenviar) trafico de datos de otros nodos y actuar como un punto de ramificación en la estructura de árbol de la subred. Un proceso de promoción satisfactorio representa la transición del estado funcional del nodo de *Terminal* a *Switch*. Es importante saber que un nodo de servicio que está en el estado *Desconectado* no puede pasar directamente al estado *Switch*.

- **Des-promoción**. Se trata del proceso por el cual un nodo de servicio deja de ser un punto de ramificación en la estructura de árbol de subred. Una des-promoción exitosa representa la transición entre el estado *Switch* y *Terminal*.

2.2.3.2 Direccionamiento

Para que dos sistemas se comuniquen se deben poder identificar y localizar entre sí. Para ello, se emplean direcciones que son básicamente etiquetas numéricas que identifican de manera lógica y jerárquica a una interfaz (elemento de comunicación/conexión) de un dispositivo dentro de una red.

Cada nodo que forma parte de una red PRIME cuenta con una dirección MAC conocida universalmente, definida en el estándar IEEE Std 802-2001, que se conoce como EUI-48. Cada una de estas direcciones es asignada a los equipos durante el proceso de fabricación y se emplea para identificar a un nodo durante el proceso de registro. El identificador EUI-48 del nodo base identifica de forma única la subred y se conoce como *Subnetwork Address (SNA)*.

Por otra parte, existen otros tipos de identificadores, como el *Local Switch Identifier (LSID)* el cual se trata de un identificador único de 8 bits que sirve para identificar a cada nodo *Switch* dentro de la subred. Este identificador es asignado por el nodo base durante el proceso de promoción. Por lo tanto, un nodo *Switch* es conocido universalmente por su identificador SNA y LSID.

El identificador *Local Node Identifier (LNID)* tiene una longitud de 14 bits y es asignado a cada nodo de servicio durante el proceso de registro. Este identificador sirve para identificar de forma única a un nodo de servicio entre todos los nodos de servicio que dependen directamente del mismo *Switch*. La combinación de los identificadores LNID y el *Switch Identifier (SID)* (que corresponde al LSID de su *Switch* más directo) forma un identificador de 22 bits conocido como *Node Identifier (NID)*. Este NID identifica a un nodo de servicio de forma única dentro de la subred.

Finalmente, durante el establecimiento de una conexión se reserva un identificador de 9 bits llamado *Local Connection Identifier (LCID)*. Este identificador LCID sirve para identificar una conexión en un nodo. La combinación de NID y LCID forma un identificador de 31 bits llamado *Connection Identifier (CID)*. Este identificador CID por su parte, sirve para identificar una conexión a nivel de subred. Por lo tanto, cualquier conexión es identificada de forma universal mediante el SNA y el CID.

La estructura completa y la longitud de los campos mencionados se muestra en la figura 2.8.

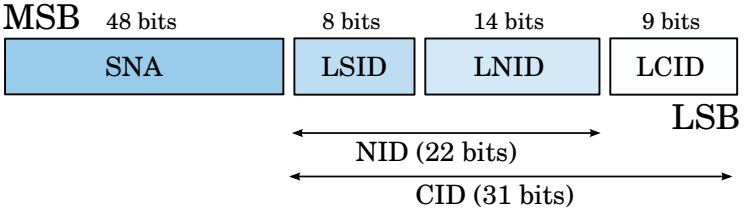


Figura 2.8: Estructura de direcciones

2.2.3.2.1 Resolución de direcciones. Cuando un nodo de servicio en el estado *Terminal* comienza con el mecanismo de promoción, el nodo base le asigna un identificador de *Switch* (LSID) único. Como ya se ha adelantado, este LSID, una vez que el nodo de servicio pase a actuar como *Switch*, será el SID de los dispositivos que se conectan a la subred a través de él. El nodo recién promocionado continúa usando el mismo NID que empleaba antes de su promoción, es decir, mantiene el SID (el LSID de su *Switch* del siguiente nivel) para hacer frente a todo el tráfico generado por o dirigido a sus procesos de aplicación locales.

Por otro lado, es importante mencionar que cada nodo de servicio tiene un nivel en la topología lógica. Así, los nodos de servicio que se conectan directamente al nodo base tiene el nivel 0, mientras que los demás nodos de servicio registrados tienen el nivel de su *Switch* inmediato más uno.

A continuación, se muestra un ejemplo de resolución de direcciones. En concreto, la figura 2.9 muestra un ejemplo en el que algunos nodos de servicio en el estado *Desconectado* están tratando de registrarse. En el ejemplo mostrado el direccionamiento tiene la siguiente nomenclatura: (SID,LNID). Al principio, el único nodo que cuenta con una dirección es el nodo base que tiene el identificador NID = (0,0).

Los demás nodos en la subred tratarán de registrarse, pero solamente los nodos A, B y C serán capaces de hacerlo y obtener su NID. La figura 2.10 muestra el estado de los nodos después del proceso de registro. Debido a que los nodos citados ya se han registrado obtendrán el SID del nodo base y un LNID único. El

nivel de los nodos recién registrados es 0 ya que están directamente conectados al nodo base.

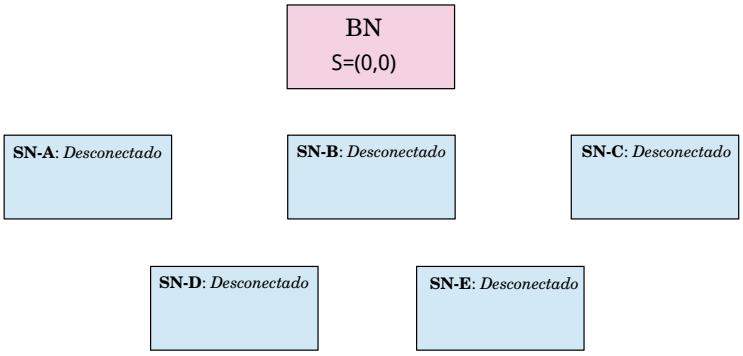


Figura 2.9: Ejemplo de resolución de direcciones: fase 1

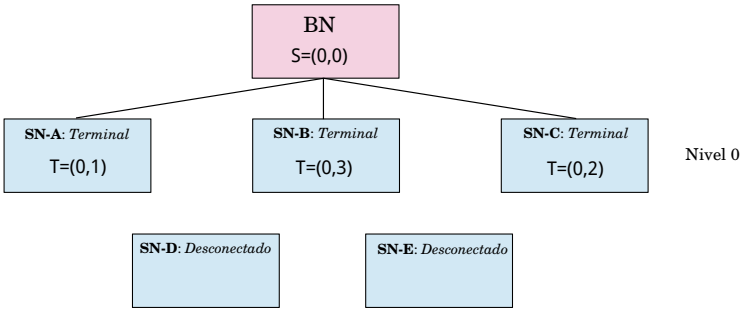


Figura 2.10: Ejemplo de resolución de direcciones: fase 2

Por otra parte, los nodos D y E no pueden conectarse directamente al nodo base, el cual es actualmente el único *Switch* de la subred. Por lo tanto, los nodos D y E enviarán paquetes PNPDU en modo *broadcast* (ver apartado 2.2.3.6.1 de este capítulo), lo que hará que los nodos en el estado *Terminal* que reciban este paquete soliciten, a su discreción, su promoción para poder extender la cobertura de la subred. Después, entre las distintas solicitudes de promoción recibidas por el nodo base, éste deberá escoger el nodo de servicio que será promocionado. En el ejemplo mostrado, el nodo B será el escogido para pasar del estado *Terminal* a *Switch*. Durante la promoción se le asignará un identificador SID único. Así, la figura 2.11 muestra el nuevo estado de la subred después de la promoción del nodo B. El nuevo *Switch* seguirá utilizando el NID que le fue asignado durante el proceso de registro para su propia comunicación como nodo *Terminal* (0,3) y el nuevo SID (1,0) se empleará para todas las funciones de *switching*.

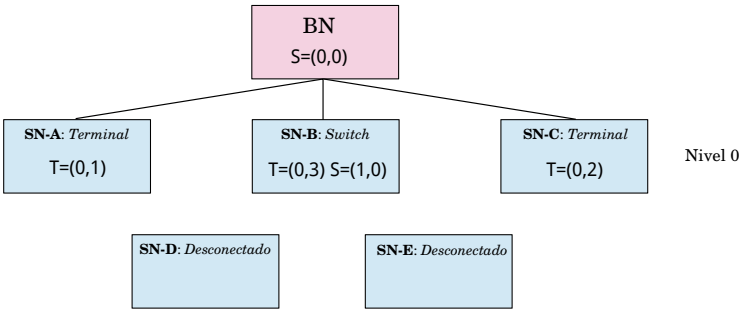


Figura 2.11: Ejemplo de resolución de direcciones: fase 3

Cuando se completa el proceso de promoción del nodo B, los nodos D y E comenzarán con el proceso de registro y obtendrán un LNID único. Cada nodo en la subred tendrá entonces un único NID para poder comunicarse como *Terminal* y los nodos *Switch* tendrán su identificador SID para realizar las funciones de *switching*. El nivel de los nodos recién registrados es 1 ya que se han registrado a través de nodos del nivel 0. Por lo tanto, una vez que todos los nodos se hayan registrado, la subred quedaría tal y como se ve en la figura 2.12.

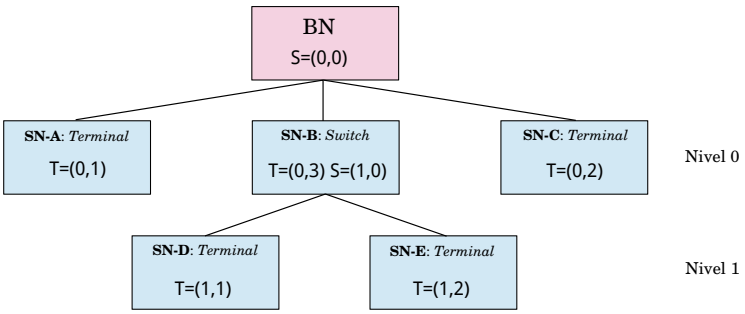


Figura 2.12: Ejemplo de resolución de direcciones: fase 4

2.2.3.3 Switching

En una subred, el nodo base no siempre puede comunicarse con todos los nodos directamente, por lo que los nodos *Switch* transmiten el tráfico desde y hacia al nodo base para que todos los nodos de la subred sean capaces de comunicarse con él. Para ello, los nodos *Switch* retransmiten tráfico que se origina o va destinada a un nodo de servicio que está en su dominio de forma selectiva y el resto del tráfico es descartado, optimizando así la fluidez de la red.

2.2.3.3.1 Tabla de *switching*. Cada nodo *Switch* mantiene una tabla de otros nodos *Switches* que están conectados a la subred a través de él. Esta información es suficiente para realizar el *switching* ya que el tráfico que va dirigido hacia o desde los nodos *Terminales* contiene el identificador del nodo *Switch* (*PKT.SID*)

a través del cual se han unido a la subred. De este modo, la función de *switching* se simplifica de manera significativa ya que no es necesario tener que mantener una lista exhaustiva de todos los nodos *Terminales* conectados a través de qué nodos.

Al principio, los nodos *Switch* comienzan con una tabla vacía sin entradas. La tabla de *Switches* se actualiza dinámicamente haciendo un seguimiento de los paquetes de promoción y des-promoción que atraviesan la red. Así, por cada paquete *PRO_ACK* (ver apartado 2.2.3.6.3) que pase por un *Switch*, este comprobará el campo *PKT.SID* de ese paquete. Si el valor de este campo coincide con el identificador del *Switch* o con alguna entrada de su tabla de *Switches*, se crea una nueva entrada en la tabla de *Switches*.

Del mismo modo, si el *Switch* reenvía una respuesta *PRO_DEM_X* (ver apartado 2.2.3.6.4) se borrará la entrada que coincida con el *Switch* que va a despromocionar de la tabla de *switching*. Sin embargo, esta entrada no se borra de manera inmediata ya que antes de hacerlo habrá que esperar *(macMaxCtlReTx + 1) * macCtlReTxTimer* número de segundos (para ver los valores de estos parámetros ir al apartado A.2.4). Este tiempo asegura que todos los paquetes que estén utilizando dicho LSID hayan abandonado la red antes de borrar la entrada de la tabla.

La figura 2.13 muestra el proceso de la formación de las tablas de *switching*. Hay que tener en cuenta que el valor del campo *PRO.NSID*, que se incluye en el paquete de respuesta *PRO_ACK*, corresponde al identificador LSID asignado por el nodo base al nuevo nodo *Switch*. Por otro lado, la figura 2.14 muestra un ejemplo de una subred en la que los *Switches* muestran sus tablas de *switching*.

2.2.3.3.2 Proceso de switching Como ya se ha adelantado, los nodos *Switches* encaminan el trafico a su dominio de una forma selectiva. Para que los datos recibidos puedan ser conmutados deberán cumplir las siguientes condiciones que se enumeran a continuación, de lo contrario, los datos serán descartados.

En primer lugar, es importante destacar que existen dos tipos de paquetes: paquetes de tipo *DOWNLINK* y *UPLINK*. Los paquetes de tipo *DOWNLINK*, tienen el campo *HDR.DO* fijado a 1 y deberán cumplir las siguientes condiciones:

- El nodo destino del paquete se ha conectado a la red a través de este nodo *Switch* (nodo *Switch* que está procesando el paquete). El *PKT.SID* es igual al SID del nodo *Switch* o su tabla de *switching* tiene una entrada con este valor.

- El paquete tiene una dirección de destino *broadcast* (el campo *PKT.LNID* es igual a $0x3FFF$) y fue enviado por el nodo *Switch* mediante el cual este nodo (nodo *Switch* que está procesando el paquete) fue registrado.

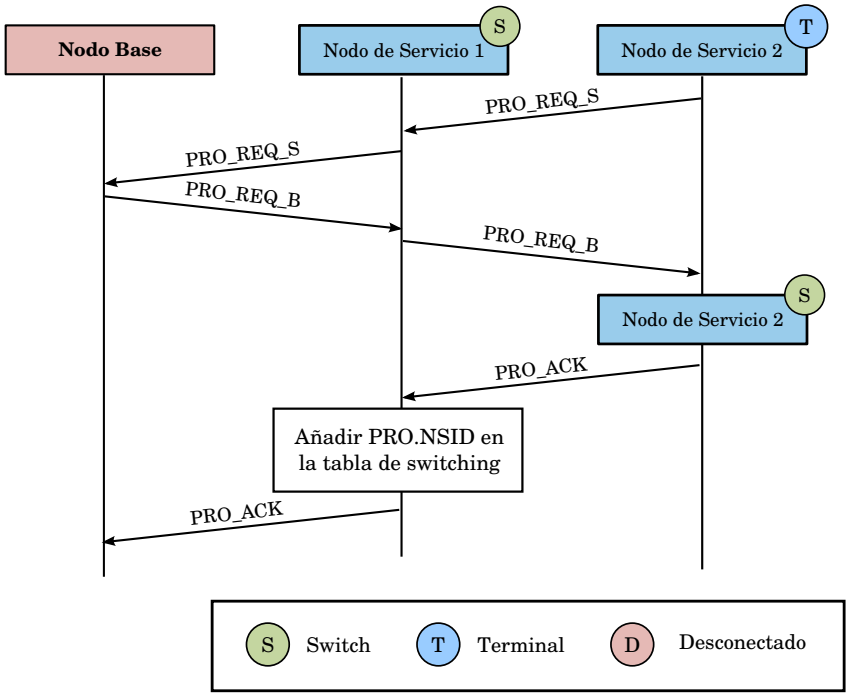


Figura 2.13: Diagrama de secuencia de la formación de las tablas de *switching*

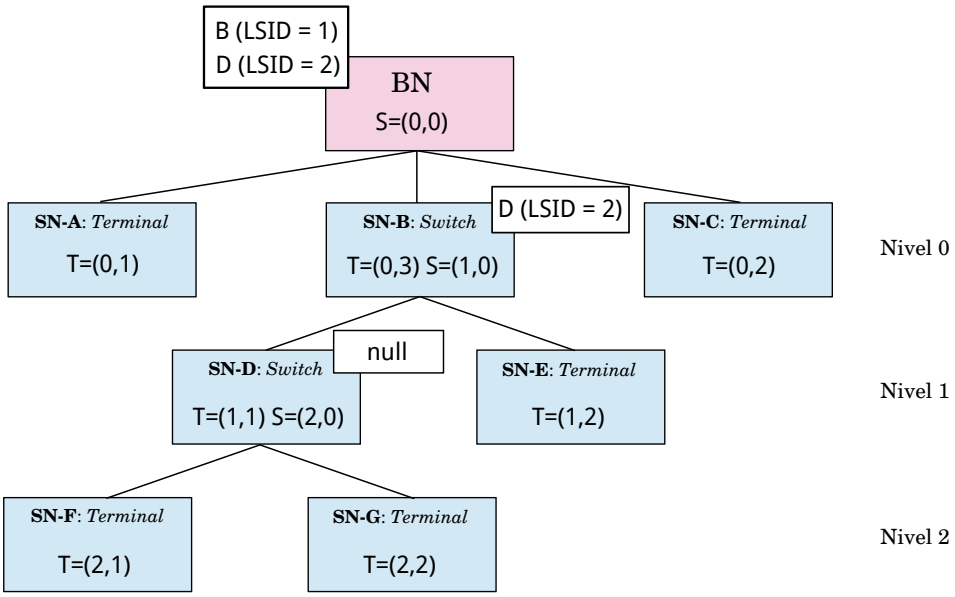


Figura 2.14: Ejemplo de tablas de *switching* en una subred

- El paquete tiene un destino *multicast* (*PKT.LNID* = $0x3FFE$), fue enviado por el nodo *Switch* mediante el cual este nodo (nodo *Switch* que está procesando el paquete) fue registrado (*PKT.SID* = SID de este nodo *Switch*) y al menos uno de los nodos de servicio conectados a la subred a través de este nodo *Switch* es miembro del grupo *multicast* al que va dirigido el paquete (indicado en el campo PKT.LCID).

Los paquetes *UPLINK* (tienen el campo *HDR.DO* fijado a 0) deberán cumplir las siguientes condiciones para que puedan ser conmutados:

- El origen del paquete está conectado a la subred a través del nodo *Switch*. El campo *PKT.SID* es igual al SID de este nodo *Switch* o su tabla de *switching* tiene una entrada con el valor del campo *PKT.SID*.

- El paquete tiene una dirección de destino *broadcast* o *multicast* (el campo *PKT.LNID* es igual a $0x3FFF$ o $0x3FFE$) y ha sido transmitido por un nodo registrado a través de este nodo *Switch* (el campo *PKT.SID* es igual al LSID de este nodo *switch*)

2.2.3.3.3 Switching de paquetes broadcast. Como se ha comentado anteriormente, los paquetes *broadcast* se distinguen porque su campo *PKT.LNID* tiene el valor $0x3FFF$.

Cuando el paquete es de enlace ascendente o *UPLINK* (*HDR.DO* = 0) el tratamiento del paquete es como si se tratase de un paquete *unicast* dirigido al nodo base. Un *Switch* que recibe un paquete de este tipo deberá aplicar las reglas explicadas en el apartado anterior.

En cambio, cuando el paquete es de enlace descendente o *DOWNLINK* (HDR.DO = 1) se transmite al siguiente nivel. Un *Switch* que recibe un paquete de este tipo deberá aplicar las reglas explicadas en el apartado anterior.

2.2.3.3.4 Switching de paquetes multicast. Para poder realizar el *switching* de paquetes *multicast*, los *Switches* deberán contar con una tabla de encaminamiento *multicast*. Dicha tabla contiene una lista de LCIDs de grupos *multicast* a los que pertenecen los miembros conectados a la subred a través de este nodo *Switch*. El LCID es el identificador de los grupos *multicast* y será incluido en los paquetes de este tipo para indicar a qué grupo pertenecen. Así, cuando llega un paquete con destino *PKT.LNID* = $0x3FFE$, si el nodo de servicio que lo recibe es un *Switch*, comprobará si el campo *PKT.LCID* del paquete coincide con alguna entrada LCID de la tabla *multicast*. Por lo tanto, el tráfico *multicast* solamente se encamina si en la tabla de encaminamiento *multicast* hay alguna entrada del LCID de los paquetes *multicast*.

Para rellenar la tabla de encaminamiento *multicast*, los *Switches* analizarán los paquetes *MUL_JOIN* y *MUL_LEAVE* que pasen por dicho *Switch*. Así, cuando

un nodo de servicio que está en el dominio de un *Switch* se une a un grupo *multicast*, el *Switch* añade una entrada con el LCID del grupo en la tabla. Asimismo, a medida que se van uniendo nodos de servicio que están en el dominio de este *Switch* al grupo *multicast*, el nodo *Switch* irá registrando los NIDs asociados a dicho LCID.

Del mismo modo, cuando un nodo de servicio que depende del *Switch* abandona el grupo, éste borra la entrada NID del nodo que acaba de abandonar el grupo. Cuando ya no queda ningún nodo de servicio asociado al grupo *multicast*, el nodo *Switch* elimina la entrada LCID de dicho grupo *multicast* de la tabla de encaminamiento.

2.2.3.4 Acceso al canal

El tiempo en los sistemas PLC PRIME se divide en unos intervalos conocidos como tramas MAC. Dentro de esta trama MAC se distinguen tres partes principales que se muestran en la figura 2.15: los *slots* para el envío de balizas, el *Shared Contention Period (SCP)* o periodo de contención compartido y el *Contention Free Period (CFP)* o periodo libre de contención.

La mayoría de las veces, los nodos de servicio y el nodo base de una subred acceden al canal durante el SCP, pero también tienen la opción de acceder durante el CFP.

Para el acceso al canal mediante CFP los dispositivos deben realizar una petición de reserva al nodo base. Dependiendo del uso del canal en esos momentos, el nodo base podrá proporcionar acceso al dispositivo que lo ha pedido durante un tiempo o denegar la petición.

El SCP en cambio, no requiere de ningún arbitraje. Sin embargo, los nodos transmisores deben respetar los limites temporales del SCP dentro de la trama MAC. La composición de una trama MAC en lo que se refiere a SCP y CFP se comunica en cada trama mediante las balizas.

Por lo tanto, una trama MAC, se compone de una o más balizas, un periodo SCP y opcionalmente un periodo CFP. Cuando el CFP está presente en la trama, la duración del mismo se indica en la baliza.

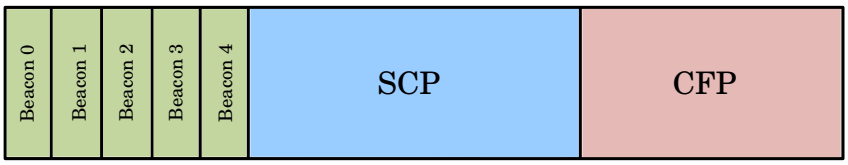


Figura 2.15: Estructura de la trama MAC

2.2.3.4.1 Balizas. Las balizas son transmitidas como tramas *Beacon PDU (BPDU)* por cada nodo *Switch* de la subred incluyendo al nodo base. El objetivo de este tipo de tramas es el de hacer circular información acerca de la estructura

de la trama MAC y por tanto, del acceso al canal para todos los dispositivos que forman parte de la subred. Los BPDU se transmiten a intervalos fijos de tiempo y también son utilizados como un mecanismo de sincronización por los nodos de servicio.

El nodo base transmite una trama BPDU (baliza) cada *MACFrameLength* - *MACBeaconLength* número de símbolos (ver apartado A.2.4), es decir, envía una baliza por cada trama MAC. Los nodos de servicio que actúan como *Switch* también envían BPDUs para mantener la parte de la subred que depende de ellos. Sin embargo, aunque estas tramas BPDU se transmiten a intervalos regulares, la frecuencia de transmisión no necesariamente tiene que ser la misma que la del nodo base, por lo que, un nodo de servicio que actúa de *Switch* no transmitirá su BPDU en cada trama.

La duración de una baliza es de *MACBeaconLength* número de símbolos. Esta longitud corresponde a la duración de la baliza sin tener en cuenta las cabeceras de la capa física. Por otro lado, debido a que los BPDUs serán recibidos por todos los nodos que estén en el dominio del *Switch* que las genera, la baliza será transmitida con el esquema de modulación más robusto (DBPSK) y con el mecanismo FEC activado. La estructura de la trama BPDU en detalle se incluye en el apartado 2.2.3.5.1 de este capítulo.

2.2.3.4.2 Beacon-slots. Como ya se ha adelantado, las tramas BPDU o balizas se transmiten a intervalos fijos de tiempo y, para ello, se emplean los llamados *beacon-slot*s. Una trama MAC puede contener *macBeaconsPerFrame* número de BPDUs (ver apartado A.2.4). Todos los *beacon-slots* se sitúan al comienzo de la trama, tal y como muestra la figura 2.15. El primer *slot* de cada trama está reservado para transmitir la baliza del nodo base. Además, debido a que los nodos *Switch* no tienen por qué enviar balizas en cada una de las tramas, el número de balizas puede cambiar de una trama a otra. Esta información, es decir, el número de balizas que hay en cada trama lo indica el nodo base en su BPDU.

Por lo tanto, cuando un nodo promociona y se convierte en *Switch*, el nodo base le asigna un *beacon-slot* para que pueda transmitir su baliza. Para ello, el nodo base le enviará un paquete de tipo *Beacon Slot Indication (BSI)* en el que se incluyen los detalles específicos del *beacon-slot* que deberá usar el nuevo *Switch*.

El algoritmo empleado por el nodo base a la hora de escoger el *beacon-slot* a asignar a un *Switch* no está definido en el estándar, por lo que cada fabricante implementa su propio algoritmo.

2.2.3.4.3 Beacon Superframes. Cuando se cambia la estructura de la trama, es decir, cuando se añade o libera un *beacon-slot*, se cambia el *beacon-slot* que debe usar un *Switch*, etc., es necesario indicar el momento en el que deberá ocurrir dicho cambio. Cuando ocurre un cambio de este tipo, es muy importante

que todos los nodos apliquen el cambio indicado por el nodo base en el mismo instante, ya que de lo contrario podría haber muchas colisiones entre balizas.

Para solucionar este problema, el estándar define el concepto de *Beacon Superframe*, donde cada baliza tiene un número de secuencia de 5 bits y, por tanto, cada *superframe* o supertrama estará compuesta por 32 tramas. De este modo, cuando el nodo base envía una notificación de un cambio, dichos cambios solamente deberán aplicarse cuando la secuencia de la baliza coincida con el número de secuencia incluido en la petición de cambio.

2.2.3.4.4 Shared-contention period Como ya se ha adelantado, el SCP es el periodo durante el cual cualquier nodo de servicio puede transmitir paquetes. Siguiendo la estructura de la trama MAC mostrada en la figura 2.15, el SCP comienza inmediatamente después del último *beacon-slot*. Debido a que este periodo de tiempo es compartido por todos los nodos, para evitar que ocurran colisiones a la hora de intentar acceder al mismo tiempo al canal, se emplea el mecanismo *Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)*.

La longitud del SCP puede variar de una trama a otra y esto viene indicado por la baliza enviada por el nodo base. En cualquier caso, la duración mínima de SCP es de al menos *MACMinSCPLength* símbolos (ver apartado A.2.4). Por otra parte, la duración máxima del SCP solamente puede darse cuando no hay ningún CFP asignado y no hay ningún nodo de servicio en el estado *Switch* en la subred (sólo hay un *beacon-slot* por trama).

Por otro lado, es importante mencionar que el uso de SCP no está limitado a las tramas en las que se reciben balizas. En las partes más profundas de la subred, el *Switch* que domina la zona puede que envíe balizas a frecuencias más bajas que el nodo base, que envía una baliza por trama. Para estas partes de la subred, en el caso de las tramas en las que no se reciben balizas, la estructura de la trama seguiría siendo la misma que en aquellas que sí se reciben.

- Algoritmo CSMA/CA. Como ya se ha adelantado, el SCP es compartido por todos los nodos de la subred, por lo que para evitar que ocurran colisiones a la hora de intentar acceder al canal al mismo tiempo, los nodos deberán implementar el mecanismo CSMA/CA que se describe en el estándar.

Como se observa en la figura 2.16 cualquier implementación del algoritmo comienza con un tiempo de *backoff* o de contención aleatorio basado en la prioridad de los paquetes a transmitir. Para ello, se definen distintos niveles de prioridad *MACPriorityLevels* (ver apartado A.2.4) en cada implementación. Cuanto más bajo sea este valor mayor será la prioridad del paquete a transmitir.

Para calcular dicho periodo de contención o *macSCPRBO* se sigue la ecuación mostrada en 2.1, donde *Priority* hace referencia al nivel de prioridad del paquete que el nodo se dispone a transmitir, *txAttempts* es el valor del número de veces

que se ha intentado transmitir dicho paquete y *macSCPLength* es la longitud del SCP de la presente trama MAC.

$$macSCPRBO = random(0,MIN(2^{Priority+txAttempts}+1),(macSCPLength/2)) \quad (2.1)$$

El resultado de esta ecuación indicará el número de símbolos que durará el periodo *backoff*. Sin embargo, antes de que comience dicho periodo, el nodo debería asegurarse de que el tiempo SCP restante de la trama es suficientemente largo como para que quepa el *backoff*, el número de comprobaciones de canal (en función de la prioridad del paquete) y la transmisión del paquete. En caso de que no sea así, el dispositivo deberá esperar al comienzo del siguiente periodo SCP de la siguiente trama. En este siguiente intento el valor del *backoff* se vuelve a calcular.

Cuando se completa el tiempo marcado por *macSCPRBO* símbolos, se lleva a cabo la comprobación de canal (detección de portadora). De este modo, debido al carácter aleatorio del periodo de *backoff*, los distintos nodos de la subred, se dispondrán a comprobar el canal en instantes diferentes. Es importante mencionar que en función de la prioridad del paquete a transmitir, la comprobación del canal se realizará una o más veces. Para calcular el número de veces que se debe realizar dicha comprobación se realiza el siguiente cálculo mostrado en la ecuación 2.2. Cuando el resultado es mayor que 1, cada comprobación de canal se realizará con una separación temporal de 3 milisegundos.

$$macSCPChSenseCount = Priority + 1 \quad (2.2)$$

Cuando el nodo comprueba que el canal está libre *macSCPChSenseCount* número de veces seguidas, éste considerará que puede proceder a la transmisión del paquete inmediatamente.

Por el contrario, si durante cualquiera de las *macSCPChSenseCount* veces el canal se detecta como ocupado, se resetean las variables, el contador $txAttempts$ incrementa en uno su valor y el proceso CSMA/CA se reinicia calculando un nuevo valor para *macSCPRBO*.

Si el algoritmo CSMA/CA se reinicia un número de *macSCPMaxTxAttempts* veces (ver apartado A.2.4), debido a que el canal se encuentra ocupado por otros paquetes, el nodo cancelará la transmisión informando a capas superiores del fallo ocurrido en el mecanismo CSMA/CA.

2.2.3.5 Tipos de paquetes

En la presente sección, se incluye una descripción de los distintos tipos de paquetes y su estructura que define el estándar.

En la parte transmisora, la capa física o PHY recibe un trama *MAC Protocol Data Unit (MPDU)* de la capa MAC y genera una trama PHY, cuya estructura se

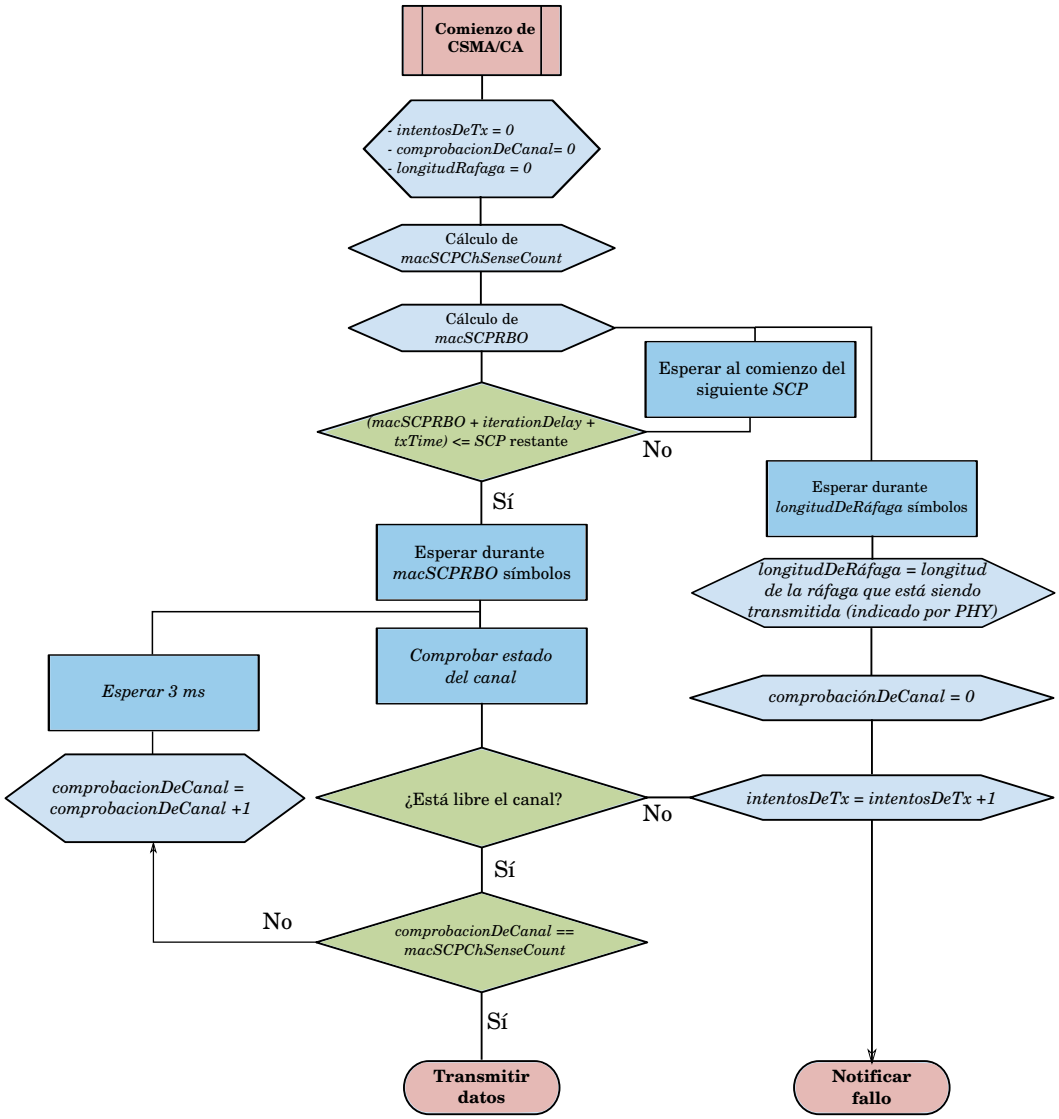


Figura 2.16: Diagrama de flujo del mecanismo CSMA/CA

detalla en el apartado 2.2.2.5 de este capítulo. A continuación, se describirá la estructura de las tramas MPDU.

2.2.3.5.1 Formato de la trama MAC PDU. Existen distintos tipos de tramas *MAC Protocol Data Unit (PDU)*s para distintos propósitos, por lo que, en esta sección, se hará una descripción de los mismos.

- Generic MAC PDU. La mayoría del tráfico de subred se compone de *Generic MAC Protocol Data Unit (GPDU)*s. Los GPDUs son empleados para transmitir tráfico de datos y para la mayoría de tráfico de control.

La composición de la trama GPDU se muestra en la figura 2.17. Como se puede observar, se compone de una cabecera MAC genérica de 3 bytes seguido de uno o más paquetes MAC y un CRC de 4 bytes al final. En la sección A.2.1 del apéndice A se muestran más detalles de la estructura de las tramas GPDU.

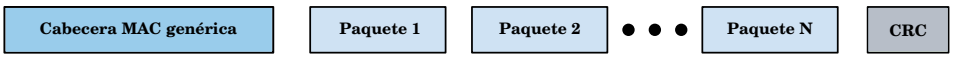


Figura 2.17: Estructura de la trama GPDU

En cuanto a la estructura de los paquetes que componen la trama GPDU, estos se componen de una cabecera de paquete (6 bytes) y un *payload* tal y como se muestra en la figura 2.18.

Figura 2.18: Estructura de paquete

- Promotion Needed PDU. Después de analizar los paquetes de tipo genérico, nos encontramos con otros tipos de tramas MAC como los *Promotion Needed PDU (PNPDU)*s que se describen en esta sección.

Cuando un nodo se encuentra en el estado *Desconectado* y no tiene conectividad con un *Switch* conocido, deberá enviar notificaciones a sus nodos vecinos para indicarles la necesidad de que un nodo en el estado *Terminal* sea promocionado. Estas notificaciones serán enviadas en forma de paquetes *Promotion Needed PDU (PNPDU)*.

Es importante mencionar que debido a que estos paquetes son enviados por nodos que aún no conocen la estructura de la trama y por tanto no saben cuándo comienza el periodo SCP, tienden a provocar colisiones, y por esta razón, tienen una cabecera reducida. La sección A.2.2 del apéndice A muestra los detalles de la estructura de este tipo de tramas.

- Beacon PDU. Como ya se ha comentado en el apartado 2.2.3.4 de este capítulo, las balizas son transmitidas en forma de tramas *Beacon PDU (BPDU)* por el nodo base y todos los *Switches* de la subred. El objetivo del envío de balizas es el de hacer circular información acerca de la estructura de la trama MAC y por lo tanto, informar acerca del acceso al medio a todos los dispositivos que son parte de la subred. Las tramas BPDU se transmiten en intervalos fijos de tiempo, por lo que también se utilizan como mecanismo de sincronización por los nodos de servicio.

Los BPDU también se utilizan para detectar cuándo un nodo *Switch* superior deja de estar disponible, ya sea por un cambio en las características del

medio, debido a un fallo, etc. Si un nodo de servicio no recibe *Nmiss-beacon* número de BPDU consecutivos (ver sección A.2.4 del apéndice A) deberá asumir que el enlace hasta su nodo *Switch* ha dejado de estar disponible. El nodo de servicio, entonces, deberá de parar de enviar balizas en el caso de que esté actuando como *Switch* y deberá cerrar todas las conexiones MAC existentes. Además, pasará al estado *Desconectado* y tratará de conectarse a una subred. Este mecanismo complementa al mecanismo de *Keep-Alive* (ver sección 2.2.3.6.8 de este capítulo) empleado por el nodo base y los *Switches* para determinar que un nodo de servicio se ha perdido.

2.2.3.5.2 Paquetes de control MAC. Una vez analizados las distintos estructuras de paquetes definidos por el estándar, en esta sección se describirán los distintos tipos de paquetes de control MAC definidos por el estándar.

Los paquetes de control MAC se transmiten como *GPDU*, por lo que el valor del campo *PKT.C* es 1.

Existen distintos tipos de paquetes de control y cada uno de ellos se identifica mediante el campo *PKT.CTYPE*. El *payload* del paquete deberá contener la información transportada por los paquetes de control y esta información será distinta en función del tipo. En la tabla 2.6 se incluye una lista con los distintos tipos de paquetes de control de la capa MAC junto con una pequeña descripción.

Tabla 2.6: Tipos de paquetes MAC de control

Tipo(PKT.CTYPE)	**Nombre**	**Descripción**
1	REG	Registro
2	CON	Conexión
3	PRO	Promoción
4	BSI	Indicación del *beacon-slot*
5	FRA	Cambio de la estructura de la trama
6	CFP	Reserva de CFP
7	ALV	Keep Alive
8	MUL	Multicast
9	PRM	Robustez de la capa PHY
10	SEC	Información de seguridad

2.2.3.6 Procedimientos MAC

Al comienzo del apartado 4.2.5.2 de descripción de la capa MAC se describen cuatro de los procedimientos más importantes de las redes PRIME que son el registro, el des-registro, la promoción y la des-promoción de los nodos de servicio. Además de estos procedimientos existen otros no menos importantes como el de la gestión de las conexiones, gestión de los grupos *multicast* y el proceso *Keep-Alive*. Todos ellos, serán descritos a continuación.

2.2.3.6.1 Mecanismo de registro. Tal y como ya se ha comentado en el apartado 2.2.3.1 de este capítulo, inicialmente, el nodo de servicio se encuentra en el estado *Desconectado*. Las únicas funciones que puede realizar en el estado *Desconectado* son recibir balizas y enviar PNPDUs. Cada nodo de servicio mantiene una tabla de *Switches* que se actualiza con la recepción de una baliza de cualquier nuevo *Switch*. En función de la implementación realizada, un nodo de servicio escogerá un nodo *Switch* de la tabla de *Switches* y procederá con el proceso de registro con el *Switch* escogido. Como ya se ha comentado, el criterio de selección del *Switch* depende de la implementación de cada fabricante y no está explicado en el estándar.

Por lo tanto, cuando un nodo de servicio se conecta a la red eléctrica se pondrá a escuchar el canal a la espera de recibir balizas durante al menos *macMinSwitchSearchTime* segundos (ver sección A.3.1 del apéndice A). Si durante ese tiempo no se ha recibido ninguna baliza de ningún *Switch*, el nodo de servicio enviará tramas *PNPDU*s en modo *broadcast* empleando el esquema de modulación más robusto (DBPSK con FEC activado) para asegurar una máxima cobertura. Además, debido a que es posible que más de un nodo de servicio esté escuchando el canal en busca de balizas al mismo tiempo, para evitar posibles colisiones entre tramas PNPDUs, opcionalmente, al valor *macMinSwitchSearchTime* se le añade un tiempo variable aleatorio. Así, los nodos que han sido conectados en el mismo instante a la red escogerán un instante distinto para el envío de sus PNPDUs. En cualquier caso, este valor añadido de tiempo no puede superar el 10 % del valor indicado por *macMinSwitchSearchTime*.

A su vez, un nodo de servicio que está buscando la promoción de uno de los nodos *Terminales* que esté en sus inmediaciones no deberá transmitir más de *macMaxPromotionPdu* número de PNPDUs por *macPromotionPduTxPeriod* segundos (ver sección A.3.1 del apéndice A). Los nodos de servicio también deben de asegurarse de que el envío de PNPDUs esté separado de forma aleatoria. Además, también deberá haber un espaciado aleatorio de tiempo entre los sucesivos envíos de PNPDUs.

Por otro lado, para no sobrecargar la red con paquetes PNPDU, siguiendo la especificación, especialmente en casos en los que se han encendido varios dispositivos al mismo tiempo, los nodos de servicio deberán reducir su tasa de envío de PNPDUs en función de un factor determinado por el numero de PNPDUs re-

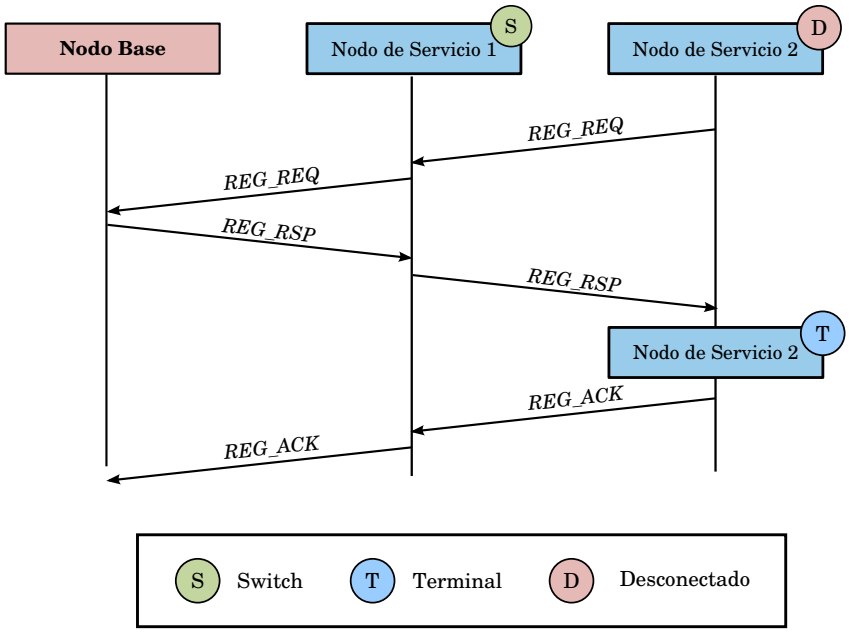


Figura 2.19: Diagrama de secuencia del proceso de registro iniciado por un nodo de servicio realizado con éxito

cibidos por otras fuentes. Por ejemplo, si un nodo recibe un PNPDU mientras está transmitiendo sus propios PNPDUs, deberá reducir sus transmisiones a no más de *macMaxPromotionPdu*/2 número de paquetes por cada *macPromotionPduTxPeriod* segundos. Igualmente, si recibe PNPDUs desde dos fuentes distintas, deberá reducir su tasa de envío a no más de *macMaxPromotionPdu*/3 número de paquetes por cada *macPromotionPduTxPeriod* segundos (ver sección A.3.1 del apéndice A).

Una vez que el nodo de servicio, en caso de necesitarlo, escoge el nodo *Switch* a través del cual comenzar el proceso de registro, envía un paquete de control del tipo *REG* al nodo base. El identificador del nodo *Switch* escogido por el nodo de servicio que está tratando de realizar el registro se indicará dentro de dicho paquete de solicitud.

En la figura 2.19 se muestra el diagrama de secuencia del proceso de registro realizado con éxito iniciado por un nodo de servicio, mientras que la figura 2.20 muestra el diagrama de secuencia de un proceso de registro rechazado por el nodo base.

2.2.3.6.2 Mecanismo de des-registro. En cualquier momento, tanto el nodo base como el nodo de servicio pueden decidir cerrar un registro existente. El nodo de servicio o nodo base que recibe una petición de des-registro deberá enviar un acuse de recibo y realizará las acciones apropiadas.

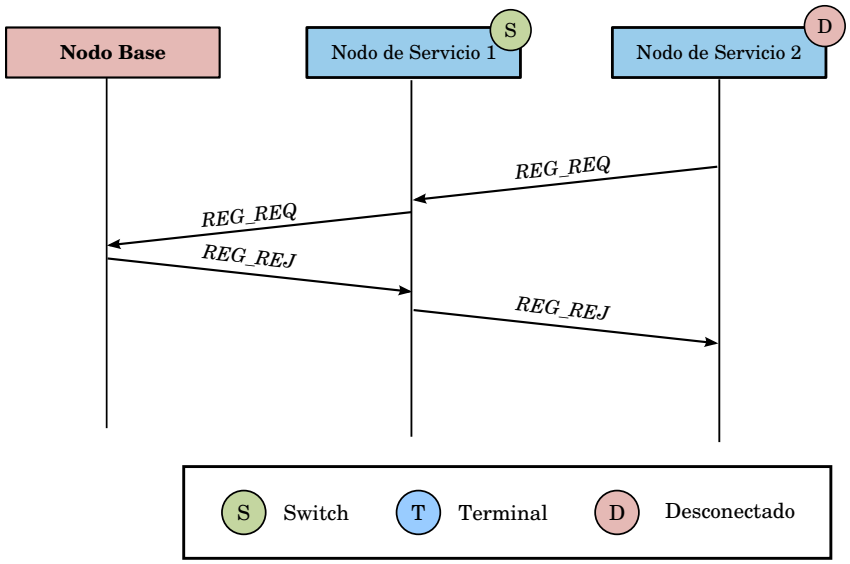


Figura 2.20: Diagrama de secuencia del proceso de registro rechazado por el nodo base

Una vez que el des-registro se haya completado, el nodo de servicio pasará de su estado funcional actual al estado de *Desconectado* y el nodo base podrá volver a reutilizar los recursos que habían sido asignados a dicho nodo.

En la figura 2.21 se muestra un proceso de des-registro iniciado por un nodo de servicio, y el mismo proceso iniciado por el nodo base.

2.2.3.6.3 Mecanismo de promoción. Cuando un nodo de servicio no consigue comunicarse con ningún *Switch* existente, enviará tramas PNPDU para que un nodo vecino *Terminal* pueda ser promocionado y se convierta en *Switch*. Durante este proceso, un nodo *Terminal* que reciba tramas PNPDUs decidirá enviar o no a su discreción, paquetes de solicitud de promoción (*PRO_REQ_S*) al nodo base.

El nodo base recogerá las peticiones de promoción enviadas por distintos *Terminales* durante un periodo de tiempo y, después de ese tiempo, procesará los paquetes y decidirá qué nodo será el que se convertirá en *Switch*. Al nodo escogido, el nodo base le enviará un paquete de aceptación de la solicitud de promoción (*PRO_REQ_B*), mientras que los demás nodos de servicio no recibirán ninguna respuesta para evitar la saturación de subred.

El nodo base también puede rechazar la petición de promoción si ocurre alguna situación especial. Por otra parte, si la subred esta preconfigurada de determinada manera, el nodo base también puede enviar solicitudes de promoción a un nodo *Terminal*.

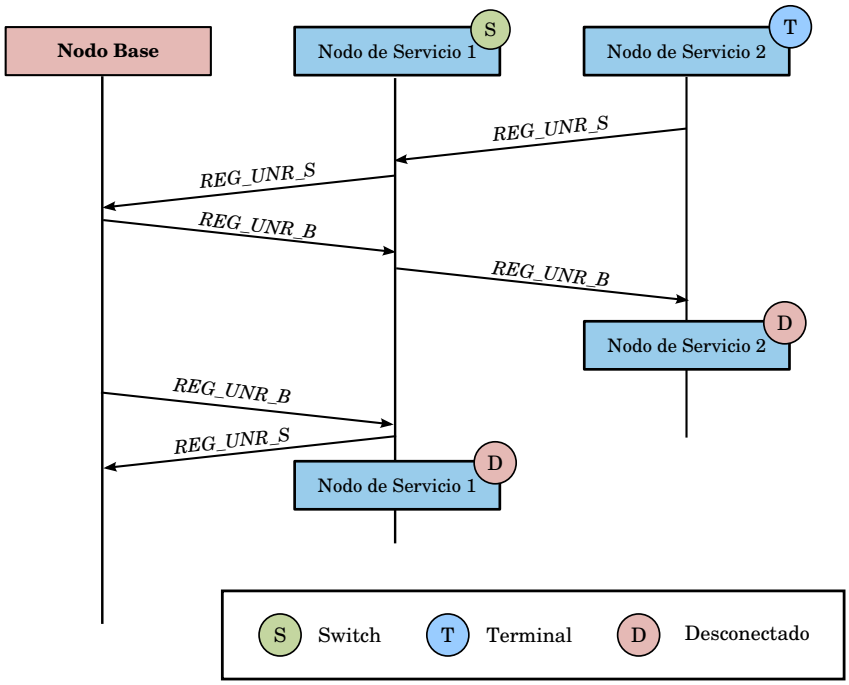


Figura 2.21: Diagrama de secuencia del proceso de des-registro iniciado por el nodo de servicio y por el nodo base

Cuando un nodo *Terminal* solicita la promoción, el campo *PRO.NSID* del paquete *PRO_REQ_S* estará puesto todo a 1s. En el caso de que la solicitud de promoción la envíe el nodo base mediante un paquete *PRO_REQ_B* el campo *PRO.NSID* contendrá el LSID asignado al nuevo *Switch*.

El recién promocionado *Switch* enviará la respuesta *PRO_ACK* con el campo *PRO.NSID* con su nuevo valor. Este último *PRO_ACK* será utilizado por los nodos *Switch* intermedios para actualizar sus tablas de *switching* como se describe en el apartado 2.2.3.3 de este capítulo.

Una vez recibida la respuesta *PRO_ACK* por el nodo base, éste le enviará un paquete de tipo *Beacon Slot Indication (BSI)* para indicarle al nuevo *Switch* la información necesaria para poder transmitir su baliza. El *Switch* por su parte contestará con un *BSI_ACK* para notificar que ya ha procesado la información recibida y comenzará a transmitir sus tramas *BPDU* siguiendo las indicaciones del nodo base.

La figura 2.22 muestra el proceso de promoción iniciado desde el nodo de servicio, mientras que la figura 2.23 muestra este mismo proceso iniciado por el nodo base.

2.2.3.6.4 Mecanismo de des-promoción. El nodo base o un nodo *Switch* pueden decidir que dicho *Switch* deje de actuar como tal. Para ello, el nodo de

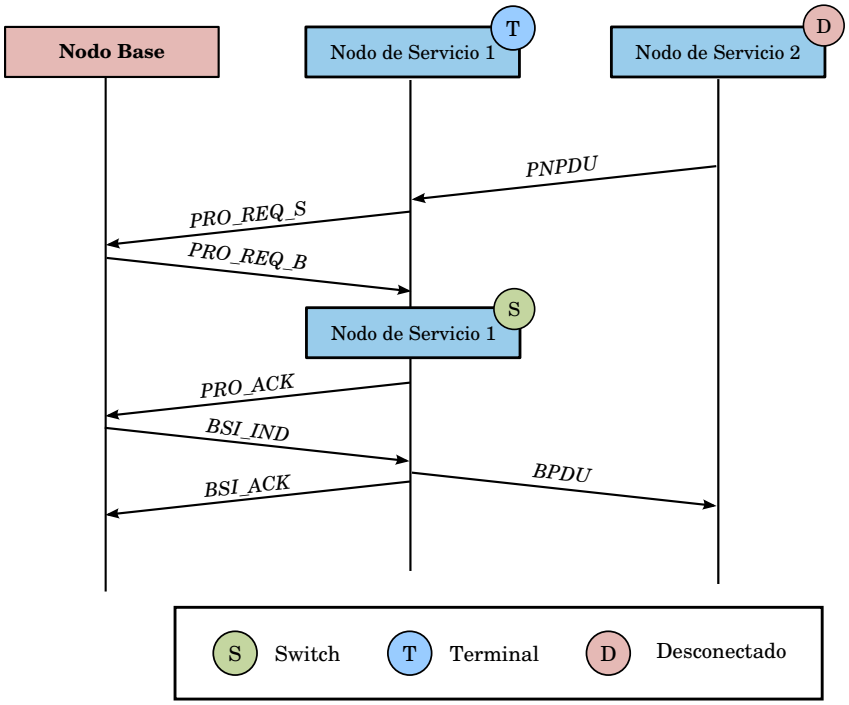


Figura 2.22: Diagrama de secuencia del proceso de promoción iniciado por un nodo de servicio

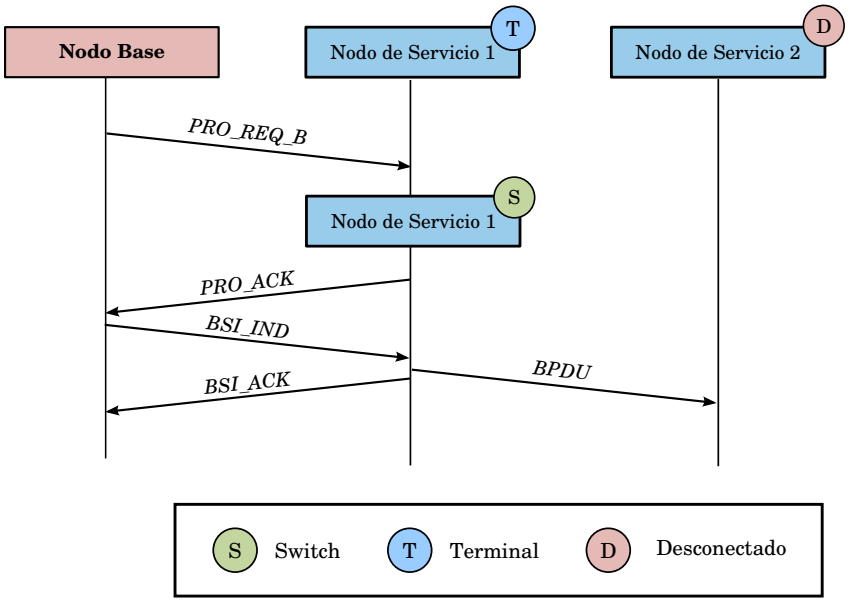


Figura 2.23: Diagrama de secuencia del proceso de promoción iniciado por el nodo base

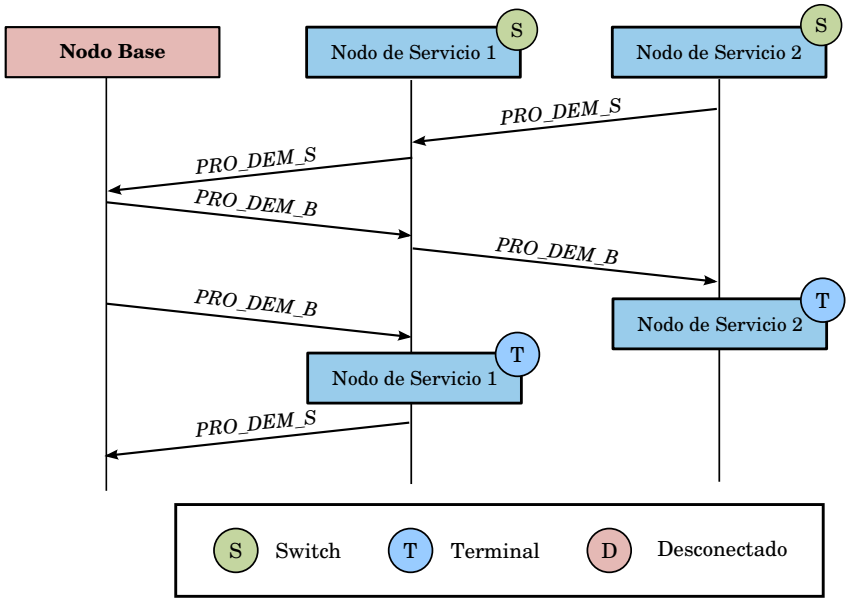


Figura 2.24: Diagrama de secuencia del proceso de des-promoción iniciado por un nodo de servicio y por el nodo base

servicio y el nodo base intercambiarán paquetes de tipo *PRO_DEM_S* y *PRO_DEM_B* tal y como muestra la figura 2.24. Después de completar el proceso de des-promoción, el nodo *Swicth* deberá parar de enviar tramas BPDU y pasará del estado *Switch* al de *Terminal*.

2.2.3.6.5 Mecanismo de conexión. El establecimiento de la conexión trabaja extremo a extremo conectando las capas de aplicación de los nodos que participan en la comunicación.

Debido a la topología en árbol de la red, la mayoría de conexiones de una subred se establecen entre el nodo base y un nodo de servicio. Sin embargo, también es posible que se abra una conexión entre nodos de servicio de la misma subred (conexiones directas).

Los paquetes de control que intervienen en el establecimiento una conexión son de tipo *CON* (*CON_REQ_S* y *CON_REQ_B*). Cuando la conexión se establece de forma satisfactoria se le asignará un LCID único para un LNID dado. La figura 2.25 muestra el diagrama de secuencia del establecimiento de conexión iniciado por el nodo base y por un nodo de servicio.

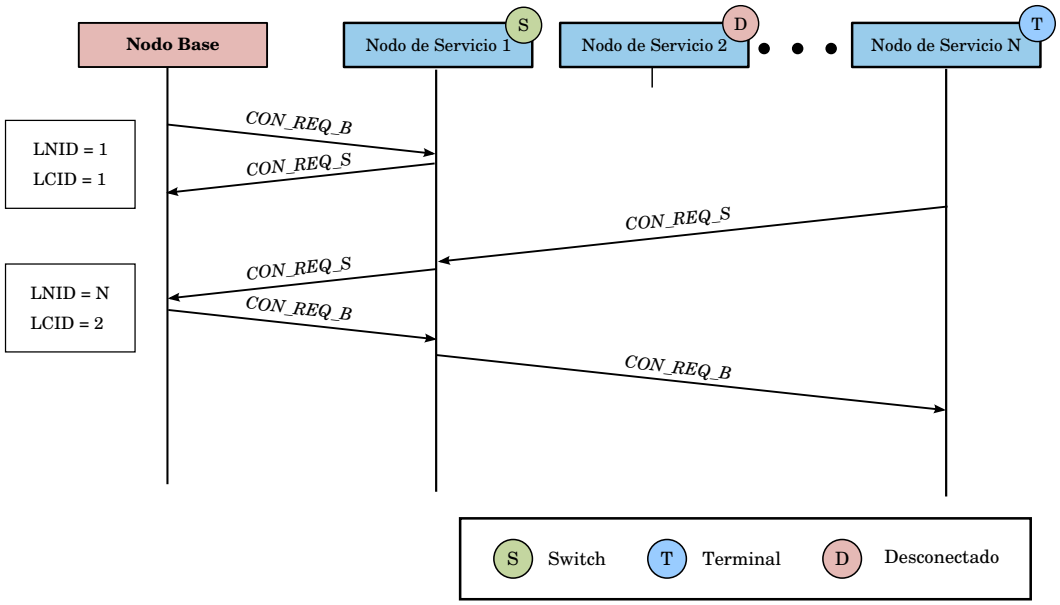


Figura 2.25: Establecimiento de conexión iniciado por el nodo base y por un nodo de servicio

2.2.3.6.6 Mecanismo de desconexión. Cualquiera de las partes de una conexión puede tomar la decisión de cerrarla en cualquier momento. Los paquetes control empleados en el cierre de una conexión son de tipo de *CON* (*CON_CLS_S* y *CON_CLS_B*). Cuando un nodo solicita el cierre de una conexión (ya sea el nodo base o nodo de servicio) la otra parte deberá enviar un paquete de respuesta como acuse de recibo para que la conexión se considere cerrada. La figura 2.26 muestra el diagrama de secuencia del cierre de la conexión iniciado por el nodo base y por un nodo de servicio.

2.2.3.6.7 Gestión de los grupo multicast. Los grupos *multicast* se utilizan para realizar el envío de información a múltiples destinos simultáneamente. Para que un nodo de servicio pueda recibir paquetes mediante este mecanismo es necesario que previamente se haya unido a un grupo *multicast*. Para llevar a cabo la gestión de este tipo de grupos (unión y abandono del grupo) se emplean paquetes de control de tipo *MUL*.

- Unión. Un nodo de servicio puede unirse a un grupo *multicast* tanto si lo solicita el propio nodo de servicio, como si lo solicita el nodo base. Los paquetes que intervienen en este proceso son *MUL_JOIN_S* y *MUL_JOIN_B*. La figura 2.27 muestra el diagrama de unión al grupo *multicast* iniciado por el nodo base y por un nodo de servicio.

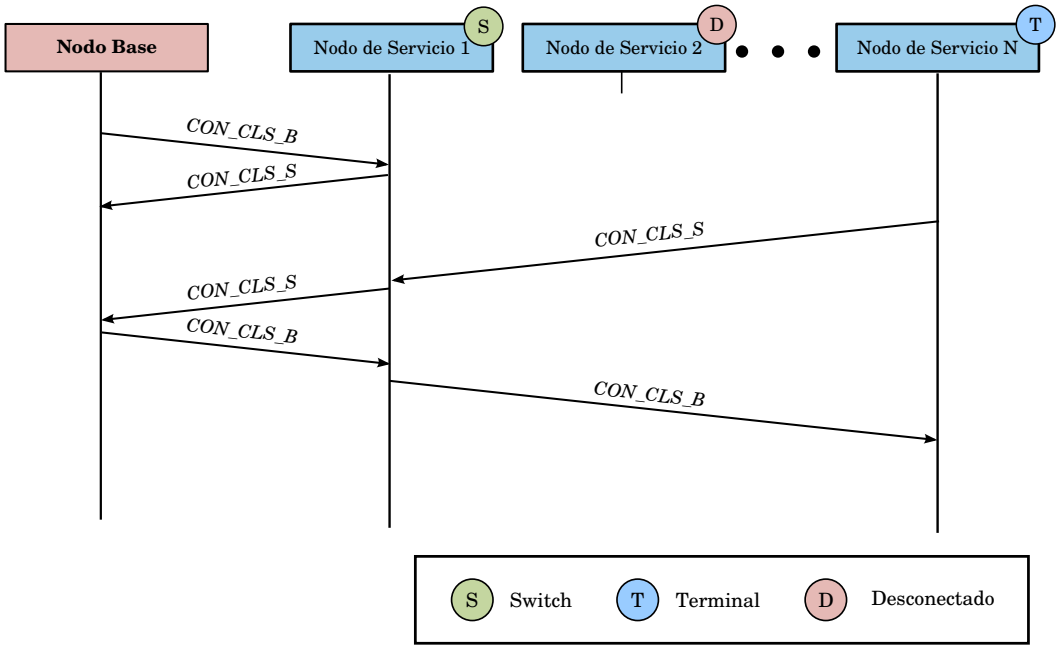


Figura 2.26: Cierre de la conexión iniciado por el nodo base y por un nodo de servicio

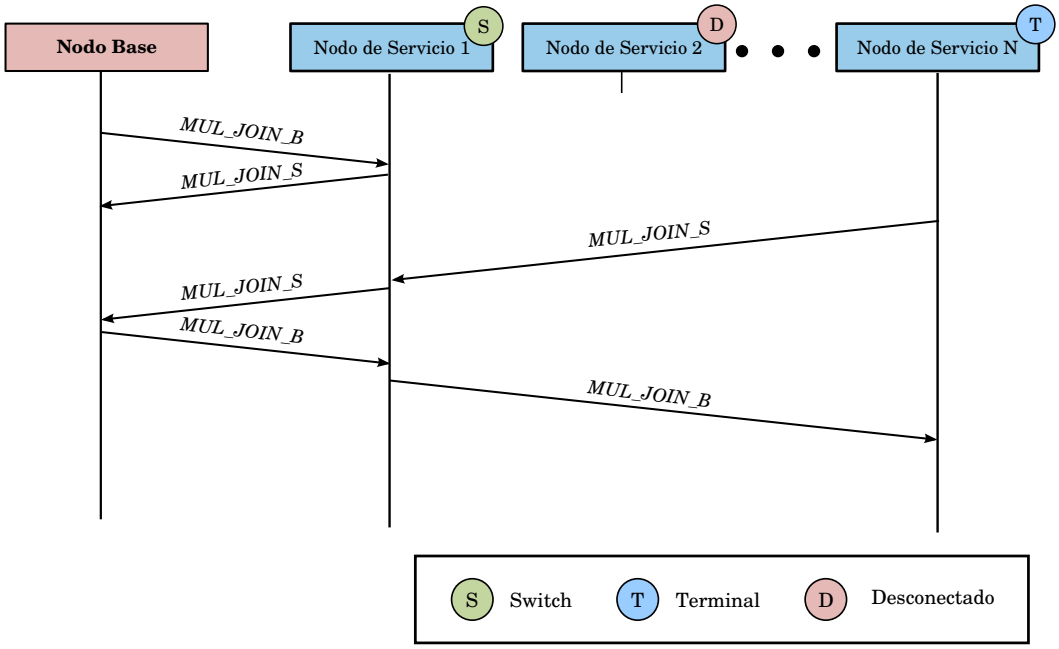


Figura 2.27: Unión al grupo *multicast* iniciado por el nodo base y por un nodo de servicio

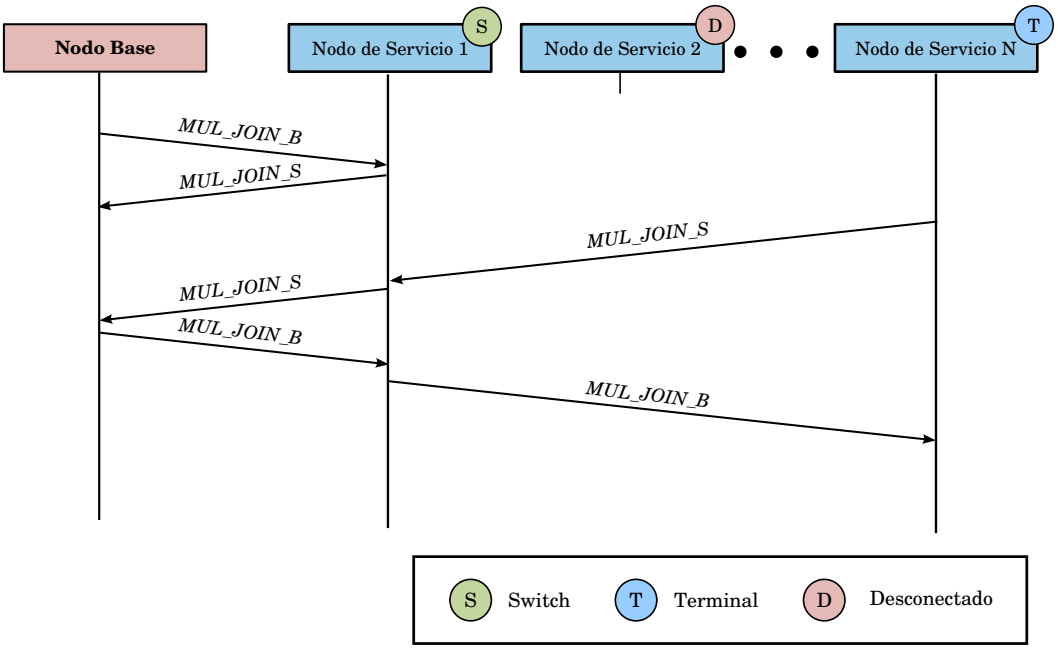


Figura 2.28: Abandono del grupo *multicast* iniciado por el nodo base y por un nodo de servicio

- Abandono. El abandono de un grupo *multicast* funciona de la misma forma que el cierre de una conexión. Este proceso puede ser iniciado tanto por el nodo de servicio como por el nodo base. Los paquetes que intervienen en este proceso son *MUL_LEAVE_S* y *MUL_LEAVE_B*. La figura 2.28 muestra el diagrama de secuencia de abandono del grupo *multicast* iniciado por el nodo base y por un nodo de servicio.

2.2.3.6.8 Proceso Keep-Alive. El proceso *Keep-Alive* es un mecanismo de la capa MAC que se utiliza para conocer si un nodo ha abandonado la subred debido a cambios en la configuración de la red o a errores de los que los nodos de servicio no se han podido recuperar. Durante este proceso, el nodo base y los nodos de servicio intercambian paquetes de tipo *ALV*. En concreto, cuando el transmisor sea el nodo base serán paquetes de tipo *ALV_B* y en caso de que sea un nodo de servicio serán de tipo *ALV_S*.

Cuando un nodo de servicio, en el proceso de registro, recibe la respuesta *REG_RSP* emplea el valor indicado en el campo *REG.TIME* para inicializar un temporizador conocido como $T_{keepAlive}$. Por cada paquete *ALV_B* que recibe el nodo de servicio, reinicia el temporizador utilizando el valor del campo *ALV.TIME* del paquete. Además, cada vez que un nodo de servicio reciba un paquete *ALV_B* desde el nodo base, le responderá con el paquete de respuesta *ALV_S*. Si el temporizador llega a su fin y no ha llegado el paquete *ALV_B*, el nodo de servicio

asume que ha sido des-registrado por el nodo base y vuelve al estado *Desconectado*. Por otra parte, la llegada de un paquete *PRO_REQ_B* (ver apartado 2.2.3.6.3) también reinicia el temporizador $T_{keepAlive}$ con el valor del campo *PRO.TIME* de dicho paquete.

Cada *Switch* intermedio por el que pasa el paquete *ALV_B*, éste guarda una copia del *PRO.TIME* y después del *ALV.TIME* por cada nodo *Switch* que está en su dominio. Así, cuando el *Switch* no recibe un paquete *ALV_S* de un nodo *Switch* que esté por debajo durante $T_{keepAlive}$ segundos definido en *PRO.TIME* y en *ALV.TIME*, el *Switch* intermedio borrará la entrada correspondiente de la tabla de *switching*.

Por otro lado, tanto el nodo base como los nodos de servicio llevan la cuenta de los paquetes *ALV_B* o *ALV_S* recibidos y transmitidos. Así, por cada paquete *ALV_B* o *ALV_S* enviado por el nodo base o nodo de servicio, el contador *ALV.TXCNT* deberá incrementar en uno su valor antes de cada envío. Por otro lado, por cada paquete *ALV_B* o *ALV_S* recibido por el nodo de servicio o el nodo base, el contador *ALV.RXCNT* será incrementado en uno. El valor de estos dos contadores se incluye en los paquetes *ALV_B* y *ALV_S*. Así, el nodo base deberá mantener un contador *ALV.TXCNT* y *ALV.RXCNT* por cada nodo de servicio. Estos contadores serán reiniciados a cero en el proceso de registro.

El estándar no define el algoritmo empleado por el nodo base para determinar con qué frecuencia enviar los paquetes *ALV_B* al nodo registrado ni cómo determinar el valor de *ALV.TIME* y *REG.TIME*. Por lo que cada fabricante implementa su propio algoritmo. La tabla 2.7 incluye los posibles valores del campo *ALV.TIME*, y por lo tanto, de los campos *REG.TIME* y *PRO.TIME*. Por otro lado, la figura 2.29 muestra el diagrama de secuencia del proceso *Keep-Alive*.

Tabla 2.7: Tiempo a esperar a que lleguen paquetes ALV_B antes de asumir que un nodo de servicio ha sido des-registrado por el nodo base.

ALV.TIME (3 bits)	Tiempo
0	32 s
1	64 s
2	128 $s \approx 2,1$ min
3	256 $s \approx 4,2$ min
4	512 $s \approx 8,5$ min
5	1024 $s \approx 17,1$ min
6	2048 $s \approx 34,1$ min
7	4096 $s \approx 68,3$ min

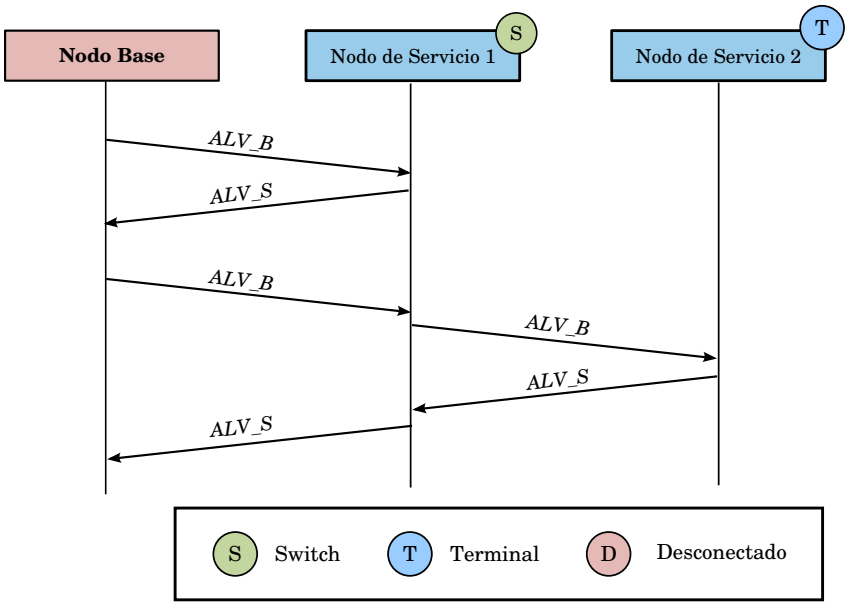


Figura 2.29: Diagrama de secuencia del proceso Keep-Alive

2.2.3.6.9 Temporizador balizas. Cuando un nodo de servicio se registra en la subred, deberá llevar la cuenta de las balizas que recibe de su nodo padre (*Switch* a través del cual se ha conectado a la subred). De este modo, si el nodo de servicio no recibe la baliza esperada $N_{miss-beacon}$ número de veces (ver apartado A.2.4 del apéndice A), el nodo de servicio considerará que su nodo padre no se encuentra disponible y por lo tanto, pasará al estado *Desconectado* y tratará de registrarse a través de otro nodo *Switch*. Este contador se reinicia cada vez que el nodo de servicio recibe la baliza esperada.

2.2.3.7 Retransmisión de paquetes de control

En el subapartado anterior se han descrito los distintos procedimientos de la capa MAC. Como se ha podido observar, en dichos procedimientos intervienen distintos tipos de paquetes de control, que en ocasiones, debido a causas como colisiones, ruido, etc. no llegan a su destino. Para la recuperación de dichos paquetes de control perdidos, el estándar define un esquema de retransmisión. El esquema de retransmisión se aplicará a los siguientes paquetes cuando requieren una respuesta:

- CON_REQ_S, CON_REQ_B;
- CON_CLS_S, CON_CLS_B;
- REG_RSP;

- PRO_REQ_B;
- BSI_IND;
- MUL_JOIN_S, MUL_JOIN_B;
- MUL_LEAVE_S, MUL_LEAVE_B;

Los dispositivos involucrados en una transacción de control de tipo MAC y que utilicen el mecanismo de retransmisión deberán contar el número de veces que retransmiten un paquete y deberán mantener un temporizador.

En la parte solicitante de una transacción de este tipo, cuando se transmite el primer paquete, el temporizador de retransmisión se inicia con el valor *macCtlReTxTimer* (ver apartado A.2.4 del apéndice A) y el contador de retransmisión se establece en 0. Si se recibe un mensaje de respuesta, el temporizador de retransmisión se detiene y la transacción se considera completa. Si el temporizador de retransmisión expira, el contador de retransmisión se incrementa. Si el contador de retransmisión es menor que *macMaxCtlReTx* (ver apartado A.2.4 del apéndice A) el paquete de control es retransmitido. Sin embargo, si el contador es igual al número máximo de retransmisiones, se devolverá un aviso informando sobre el fallo a la entidad que ha realizado la llamada. Además, si la retransmisión se lleva a cabo por el nodo de servicio, el dispositivo deberá volver al estado *Desconectado*.

En la parte de la respuesta de la transacción, el receptor del paquete deberá determinar si se trata de un paquete retransmitido o no. En cualquier caso, el nodo receptor contestará a dicho paquete con su correspondiente paquete de respuesta.

Por otra parte, para las transacciones en las que se intercambian tres paquetes, por ejemplo, en el mecanismo de promoción, como se ha mostrado en el apartado 2.2.3.6.3, la parte que responde debe realizar retransmisiones exactamente de la misma manera que la solicitante. Esto asegura que si se pierde el tercer paquete de la transacción, el paquete se volverá a enviar y la transacción será completada.

2.2.4 Nivel de gestión

En esta sección se describen los aspectos más importantes del nivel de gestión. En esta versión de la especificación 1.3.6 el nivel de gestión incluye funciones para la gestión del nodo y la actualización del *firmware* (figura 2.30).

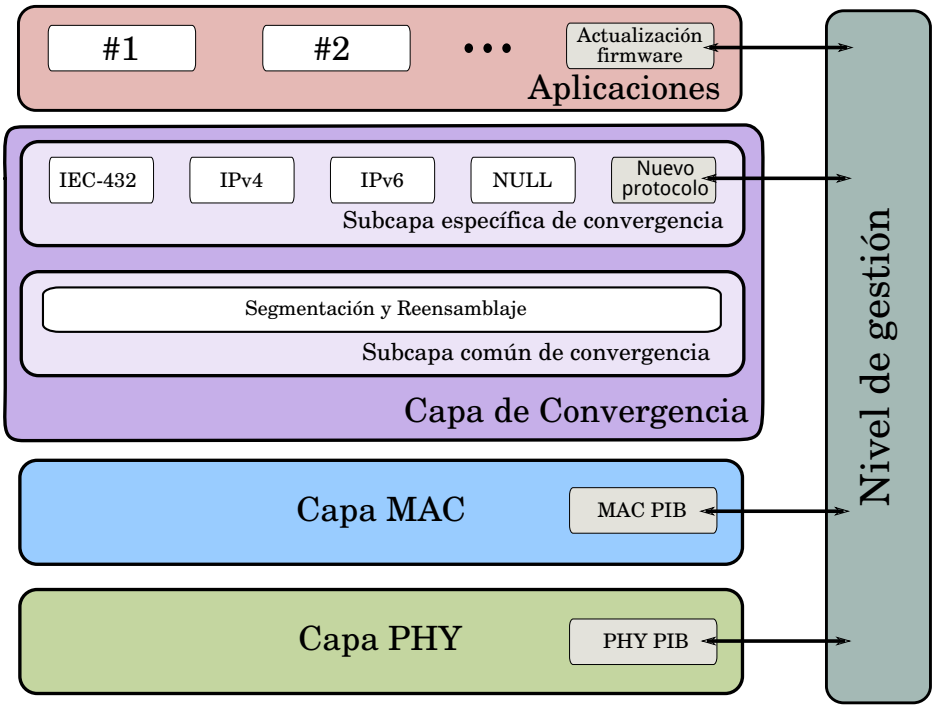


Figura 2.30: Nivel de gestión

2.2.4.1 Gestión del nodo

La gestión del nodo se lleva a cabo a través de un conjunto de atributos que se definen tanto para la capa PHY como para la capa MAC. Al conjunto de estos atributos de gestión se les conoce como *PLC Information Base (PIB)*.

2.2.4.1.1 Atributos PIB de la capa PHY. Dentro de los atributos PIB de la capa PHY se distinguen dos tipos:

- *Atributos estadísticos.* La capa PHY puede proporcionar información estadística para propósitos de gestión.
- *Atributos de implementación.* Son parámetros de sólo lectura de la capa PHY que proporcionan información de la implementación específica.

2.2.4.1.2 Atributos PIB de la capa MAC. Dentro de los atributos PIB de la capa MAC existen distintas clases de atributos que se enumeran a continuación. En la sección A.3.1 del apéndice se incluyen algunos de los atributos PIB de la capa MAC que se han ido citando a lo largo de este capítulo.

- *Atributos variables MAC.* Los atributos PIB de la capa MAC incluyen un conjunto de atributos que influyen al comportamiento funcional de una

implementación. El valor de estos atributos se puede cambiar desde una entidad externa a la capa MAC, aún incluso durante su funcionamiento.

- *Atributos funcionales.* Estos atributos de sólo lectura para las entidades de gestión informan acerca del comportamiento funcional de la capa MAC.
- *Atributos estadísticos.* La capa MAC deberá proporcionar información estadística para fines de gestión.
- *Atributos lista MAC.* Estos atributos consisten en unas listas de sólo lectura que contienen información para las entidades de gestión.
- *Atributos PIB de acciones.* Algunas de las pruebas de conformidad requieren desencadenar ciertas acciones en los nodos de servicio. Para ello, se definen un conjunto de atributos que deberán ser soportados por todas las implementaciones.

2.2.4.1.3 Atributos PIB de Aplicación. Por otro lado, existen una serie de atributos PIB de nivel de aplicación que aunque no afectan a las comunicaciones pueden facilitar la gestión de la red. Estos atributos deberán ser soportados tanto por el nodo base como por los nodos de servicio y se enumeran a continuación:

- *AppFwVersion*. Descripción de la versión de *firmware* del dispositivo.
- *AppVendorId*. Identificador de proveedor único asignado por la PRIME Alliance.
- *AppProductId*. Identificador de producto único asignado por el fabricante.

2.2.4.2 Proceso de actualización

Los equipos de telemedida que forman parte de una subred PRIME pueden contener uno o más de un *firmware* que cada cierto tiempo puede que sea necesario actualizar tanto para corregir errores como para añadir nuevas funcionalidades. Para ello, el estándar PRIME define el mecanismo de actualización *firmware* que se explicará a continuación.

2.2.4.2.1 Descripción general. El mecanismo de actualización *firmware* es capaz de funcionar en modo *unicast* o *multicast*. Todos los paquetes de control se envían empleando conexiones *unicast*, mientras que los datos pueden ser enviados en modo *unicast* (modo por defecto) o *multicast* (solamente si el fabricante lo soporta). Hay que tener en cuenta que, con el fin de asegurar la correcta recepción del *firmware* cuando se están actualizando nodos de servicio de distintos fabricantes, los paquetes de datos nunca deberán enviarse en

modo *broadcast*. Por lo tanto, solamente estará permitido trabajar en modo *unicast* o *multicast*. Por otra parte, los nodos servicio solamente contestarán a los mensajes recibidos en modo *unicast*.

Tanto las conexiones *unicast* como *multicast* son establecidas por el nodo base. En caso de que los nodos soporten el modo *multicast*, el nodo base solicitará a los nodos de un fabricante concreto que se unan a un grupo *multicast* concreto que se creará exclusivamente para llevar a cabo el proceso de actualización y será eliminado al finalizar dicho proceso.

Más tarde, tras completar la descarga del *firmware*, el nodo base ordenará a cada nodo de servicio que realice una comprobación de integridad de la imagen del *firmware* recibida. Así, si la imagen resulta estar corrupta, la descarga del *firmware* será reanudada. En cambio, si la imagen recibida es correcta, el nodo de servicio deberá esperar a que el nodo base le ordene a que ejecute el nuevo *firmware*.

El mecanismo de actualización *firmware* puede fijar el instante en el que el *firmware* recién descargado sea ejecutado por lo nodos de servicio. Por lo tanto, el nodo base puede escoger que todos los nodos se reinicien al mismo tiempo o en varios pasos.

Después del reinicio, cada nodo de servicio ejecutará el nuevo *firmware* por el tiempo especificado por el mecanismo de actualización *firmware*. Si este tiempo termina sin la recepción de la confirmación por parte del nodo base, o el nodo base decide abortar el proceso de actualización, los nodos de servicio descartarán el nuevo *firmware* y volverán a la versión antigua del mismo. En cambio, si se recibe el mensaje de confirmación, el nodo de servicio considerará que la nueva versión es la única versión valida y borrará la versión antigua.

Por lo tanto, los nodos de servicio no podrán descartar ningún *firmware* almacenado hasta que reciba el mensaje de confirmación final o hasta que el temporizador de seguridad o *SafetyTimer* llegue a su fin.

2.2.4.2.2 Segmentación. Para transferir la imagen del *firmware*, debido su tamaño (*ImageSize*), es necesario segmentarla en elementos más pequeños llamados páginas. El tamaño de las páginas (*PageSize*) puede ser variable y puede tomar uno de los siguientes valores: 32, 64, 128 o 192 bytes. Por lo tanto, teniendo en cuenta este dato, el número de páginas del *firmware* (*PageCount*) se calcula mediante la ecuación 2.3:

$$PageCount = \frac{ImageSize}{PageSize} + 1 \tag{2.3}$$

Todas las páginas tendrán el tamaño indicado por el parámetro *PageSize* excepto la última que contendrá los bytes restantes. Este parámetro es configurado por el nodo base y notificado durante la inicialización del proceso de actualización *firmware*.

2.2.4.2.3 Máquina de estados. Cuando un nodo de servicio está ejecutando el servicio de actualización *firmware* puede encontrarse en uno de los siguientes cinco posibles estados: *Idle*, *Receiving*, *Complete*, *Countdown* y *Upgrade*.

En la figura 2.31 se muestra la máquina de estados del proceso de actualización *firmware*.

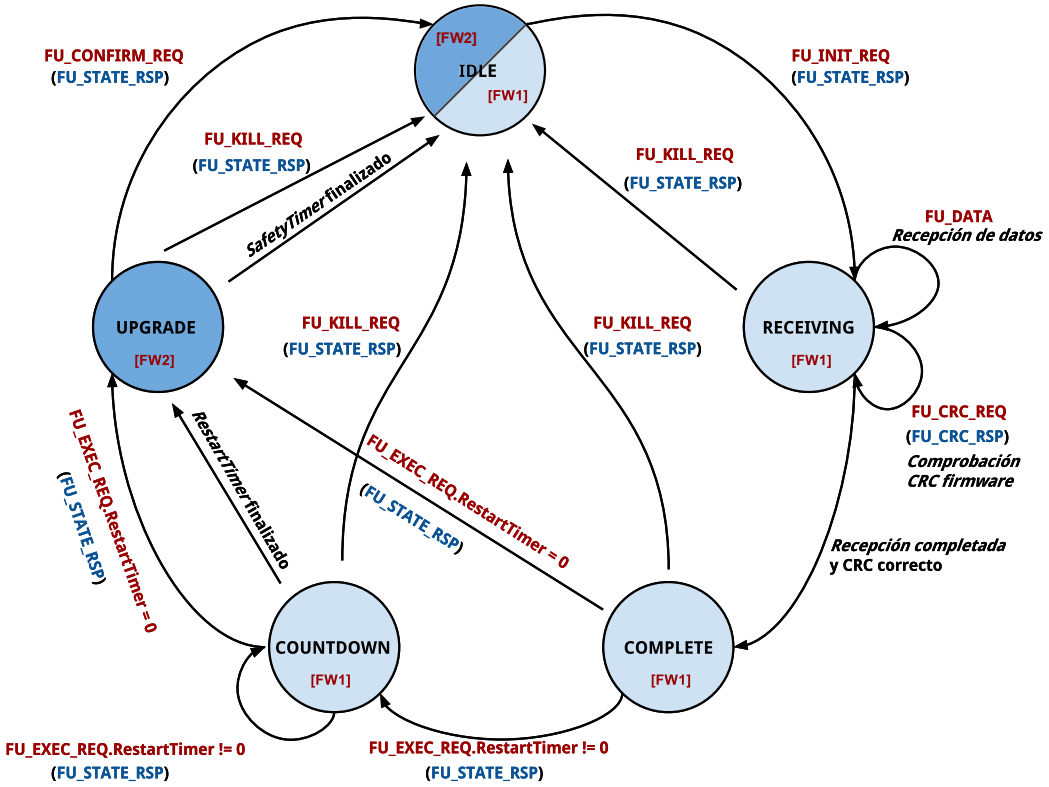


Figura 2.31: Máquina de estados del proceso de actualización *firmware*

- ***Idle***. Los nodos de servicio se encuentran en el estado "Idle" cuando no están haciendo nada. La recepción de un mensaje del tipo *FU_INIT_REQ* es el único evento que fuerza al nodo de servicio que cambie su estado al de "Receiving". Por otra parte, la llegada de un paquete del tipo *FU_KILL_REQ* forzará que el proceso de actualización termine y hará que los nodos de servicio pasen de cualquier estado al de "Idle".

- ***Receiving***. Cuando los nodos de servicio se encuentran en este estado podrán recibir las páginas de la imagen del *firmware* mediante paquetes del tipo *FU_DATA*. Una vez de que la descarga se haya completado, el nodo base le enviará al nodo de servicio un paquete del tipo *FU_CRC_REQ* para que dicho nodo de servicio realice la comprobación de la integridad del *firmware* y éste le responderá con un *FU_CRC_RSP*. La realización de este

CRC sobre el *firmware* es obligatoria y si el resultado es satisfactorio, el nodo de servicio contestará con el paquete *FU_CRC_RSP* y pasará al estado de "Complete". Sin embargo, si el CRC no resulta bien, el nodo de servicio contestará también con un paquete de tipo *FU_CRC_RSP*, descartará la imagen completa del *firmware*, actualizará su *bitmap* y esperará a la retransmisión de paquetes.

- ***Complete***. Un nodo de servicio en el estado "Complete" queda a la espera de la recepción de un paquete *FU_EXEC_REQ*. Este paquete contiene un campo llamado *RestartTimer* que le indica al nodo de servicio el instante en el que debe reiniciarse y comenzar a utilizar el nuevo *firmware*. Por lo tanto, en función del valor de dicho parámetro, el nodo de servicio pasará a uno de los siguientes estados : "Countdown" o "Upgrade". Si el valor de *RestartTimer* es cero, el nodo de servicio pasará directamente al estado "Upgrade", mientras que si es mayor que cero pasará al estado "Countdown".

- ***Countdown***. Un nodo de servicio que se encuentra en el estado "Countdown" esperará el tiempo indicado por el campo *RestartTimer* del paquete *FU_EXEC_REQ*. Transcurrido este tiempo el nodo pasará al estado "Upgrade".

- ***Upgrade***. Un nodo de servicio en el estado "Upgrade" utilizará el nuevo *firmware* durante el tiempo especificado por el campo *SafetyTimer* del paquete *FU_EXEC_REQ*. Si durante este tiempo el nodo de servicio no recibe ningún paquete de confirmación por parte del nodo base (o si recibe un paquete *FU_KILL_REQ*), el nodo de servicio descartará el nuevo *firmware*, se reiniciará con la versión antigua y cambiará al estado de "Idle".

2.2.4.2.4 Paquetes de control. En los siguientes subapartados se explicará con más detalle cada uno de los paquetes de control que intervienen en el proceso de actualización *firmware*. La estructura de los paquetes que se describen a continuación se incluye en el apartado A.3.2 del apéndice A.

- FU_INIT_REQ. El nodo base envía este paquete para configurar a un nodo de servicio para la actualización *firmware*. Si el nodo de servicio se encuentra en el estado "Idle", al recibir este paquete pasará a estar en el estado "Receiving" y contestará con un paquete *FU_STATE_RSP*. Si el nodo de servicio recibe un paquete de este tipo cuando está en cualquier otro estado simplemente contestará con el paquete *FU_STATE_RSP*.

- FU_EXEC_REQ. Este paquete es utilizado por el nodo base para ordenar a un nodo de servicio que se encuentra en el estado "Complete" que se reinicie

y comience a utilizar el nuevo *firmware* una vez de que este haya recibido la imagen al completo. El paquete *FU_EXEC_REQ* especifica el instante en el que el nodo de servicio debe reiniciarse (mediante el campo *RestartTimer*) y comenzar a utilizar el nuevo *firmware*. A su vez, este paquete también indica el valor del periodo de seguridad (*SafetyTimer*).

Además, este paquete también puede ser empleado cuando el nodo de servicio se encuentra en el estado "Countdown" para resetear tanto el *RestartTimer* como el *SafetyTimer*.

Como ya se ha comentado en apartados anteriores, dependiendo del valor del *RestartTimer*, un nodo de servicio en el estado "Complete" puede pasar al estado "Countdown" o al estado "Upgrade". En cualquier caso, el nodo de servicio contestará con un paquete *FU_STATE_RSP*.

- FU_CONFIRM_REQ. Este paquete es enviado por el nodo base a un nodo de servicio en el estado "Upgrade" para confirmar la actual versión del *firmware*. Si el nodo de servicio recibe este mensaje, descartará la versión antigua del *firmware* y pasará al estado "Idle". Asimismo, el nodo de servicio contestará con un paquete *FU_STATE_RSP* al recibir este paquete. En cambio, si el nodo de servicio se encuentra en cualquier otro estado al recibir este paquete, el nodo de servicio contestará con un *FU_STATE_RSP* sin realizar ninguna otra acción.

- FU_STATE_REQ. Este paquete es enviado por el nodo base para poder obtener información acerca del estado de la actualización *firmware* de un nodo de servicio. El nodo de servicio, cuando recibe este paquete contestará con un *FU_STATE_RSP*.

- FU_KILL_REQ. El nodo base envía este mensaje para finalizar el proceso de actualización *firmware*. Así cuando un nodo de servicio reciba este paquete, pasará automáticamente al estado "Idle" y de forma opcional eliminará los datos que haya descargado hasta el momento. Además, el nodo de servicio contestará con un paquete *FU_STATE_RSP*.

- FU_STATE_RSP. Este paquete es enviado por un nodo de servicio en respuesta a cualquiera de los siguientes paquetes: *FU_STATE_REQ*, *FU_KILL_REQ*, *FU_EXEC_REQ*, *FU_CONFIRM_REQ* o *FU_INIT_REQ* recibidos a través de conexiones *unicast*.

El paquete *FU_STATE_RSP* se emplea para notificar el estado del proceso de actualización *firmware* de un nodo de servicio. Además, este paquete también se utiliza como respuesta por defecto a todos los eventos que ocurren en estados que no están contemplados como por ejemplo: recepción de *FU_EXEC_REQ* en el estado "Receiving", recepción de *FU_INIT_REQ* en el estado "Upgrade"...

- FU_DATA. Este paquete es enviado por el nodo base para transmitir cada página de la imagen del *firmware* a los nodos de servicio. Cuando el nodo base envía este paquete, no espera ningún tipo de respuesta por parte del nodo de servicio.

- FU_MISS_REQ. Este paquete lo envía el nodo base a un nodo de servicio para pedirle información sobre las páginas que aún no ha recibido. Si el nodo de servicio se encuentra en el estado "Receiving" responderá con un paquete *FU_MISS_BITMAP* o *FU_MISS_LIST*. Si el nodo de servicio se encuentra en cualquier otro estado responderá con un *FU_STATE_RSP*.

- FU_MISS_BITMAP. Los nodos de servicio envían este tipo de paquete en respuesta a un paquete de tipo *FU_MISS_REQ* y transporta información de las páginas que aún le quedan por recibir al nodo de servicio. Por otro lado, la decisión de enviar un paquete de este tipo o uno de tipo *FU_MISS_LIST* es del propio nodo de servicio. No obstante, cuando el número de páginas que faltan es elevado se opta por el envío de paquetes *FU_MISS_BITMAP*, mientras que en el caso contrario envían paquetes de tipo *FU_MISS_LIST*. Aún así, dependiendo de la implementación de PRIME (en función del fabricante) se transmitirá un paquete u otro y el nodo base deberá ser capaz de entender ambos tipos.

- FU_MISS_LIST. Los nodos de servicio envían este tipo de paquete en respuesta a un paquete *FU_MISS_REQ* y transporta información de las páginas que aún le quedan por recibir. Como se ha adelantado, en función de la implementación de PRIME, el nodo de servicio enviará un paquete de tipo *FU_MISS_LIST* o de tipo *FU_MISS_BITMAP*.

- FU_INFO_REQ. El nodo base envía este paquete a un nodo de servicio para solicitarle información relativa al fabricante, modelo del dispositivo, versión del *firmware* y otros parámetros especificados por el fabricante. El nodo de servicio por su parte, contestará con uno o más paquetes de tipo *FU_INFO_RSP*.

- FU_INFO_RSP. Un nodo de servicio envía este paquete en respuesta a un paquete de tipo *FU_INFO_REQ* enviado por el nodo base. Es posible que el nodo de servicio tenga que enviar más de un paquete de tipo *FU_INFO_RSP* a la hora de contestar a una solicitud de información enviada por el nodo base.

- FU_CRC_REQ. El paquete *FU_CRC_REQ* lo envía el nodo base para ordenarle al nodo de servicio que realice la operación CRC sobre la imagen completa del *firmware*. El campo CRC del paquete *FU_CRC_REQ* es el mismo que el indicado por el paquete *FU_INIT_REQ*.

El nodo de servicio por su parte, contesta con un paquete *FU_CRC_RSP* en caso de que se encuentre en el estado "Receiving", mientras que si se encuentra en cualquier otro estado contestará con un *FU_STATE_RSP*. Es importante mencionar que el nodo base no deberá enviar un paquete de este tipo si la descarga del *firmware* aún no ha sido completada, o lo que es lo mismo, el *bitmap* no está completo. Si el nodo base tiene un comportamiento no esperado y envía un paquete de tipo *FU_CRC_REQ* antes de que la descarga haya terminado, el nodo de servicio contestará con un *FU_STATE_RSP*.

- FU_CRC_RSP. Es el paquete enviado por un nodo de servicio como respuesta al paquete *FU_CRC_REQ* enviado por el nodo base.

2.2.4.2.5 Ejemplos del proceso de actualización. Los ejemplos mostrados a continuación, muestran el tráfico generado entre el nodo base y el nodo de servicio durante el proceso de actualización, tanto para el caso *multicast* como para el *unicast*.

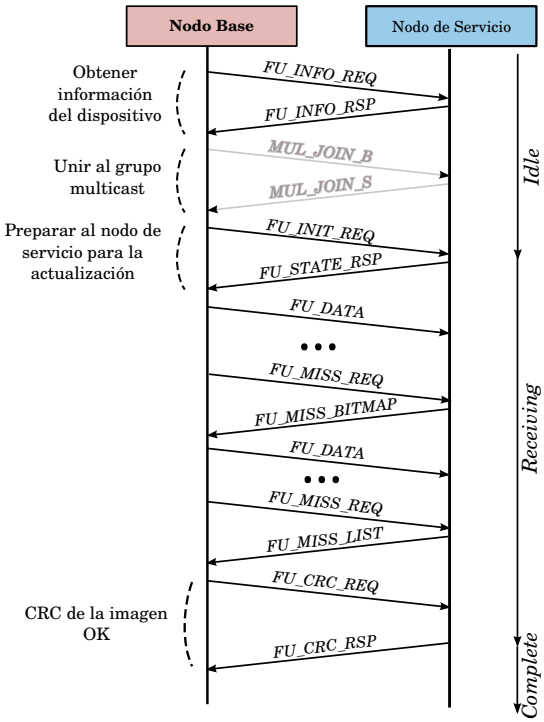


Figura 2.32: Fases de inicialización, descarga y comprobación de la imagen del *firmware*

La figura 2.32 muestra la inicialización del proceso de actualización *firmware*, la descarga de páginas y la comprobación de integridad de la imagen descargada. En el ejemplo, la descarga de la imagen del *firmware* se completa

justo antes del envío del último paquete *FU_MISS_REQ*. El nodo base envía dicho paquete para que el nodo de servicio compruebe si ya ha recibido todas las páginas. En caso de ser así, tal y como ocurre en el ejemplo, el nodo de servicio responde con un paquete *FU_MISS_LIST* cuyo campo *PageIndexList* está vacío, lo que significa que la imagen del *firmware* ya ha sido descargada por completo.

Por otra parte, la figura 2.33 muestra el modo a proceder después de completar la descarga de la imagen del *firmware*. El nodo base le ordena al nodo de servicio que se reinicie inmediatamente (Reinicio inmediato", *RestartTimer*= 0) o después de un periodo de tiempo definido (Reinicio retardado", *RestartTimer* != 0). Después del reinicio, el nodo base podrá enviar un paquete de confirmación (*FU_CONFIRM_REQ*) del *firmware* recién descargado al nodo de servicio o también le podrá enviar el paquete *FU_KILL_REQ* para que el *firmware* sea descartado. En caso de que el nodo de servicio no reciba ningún paquete y el temporizador de seguridad *SafetyTimer* llegue a su fin, el *firmware* también será descartado.

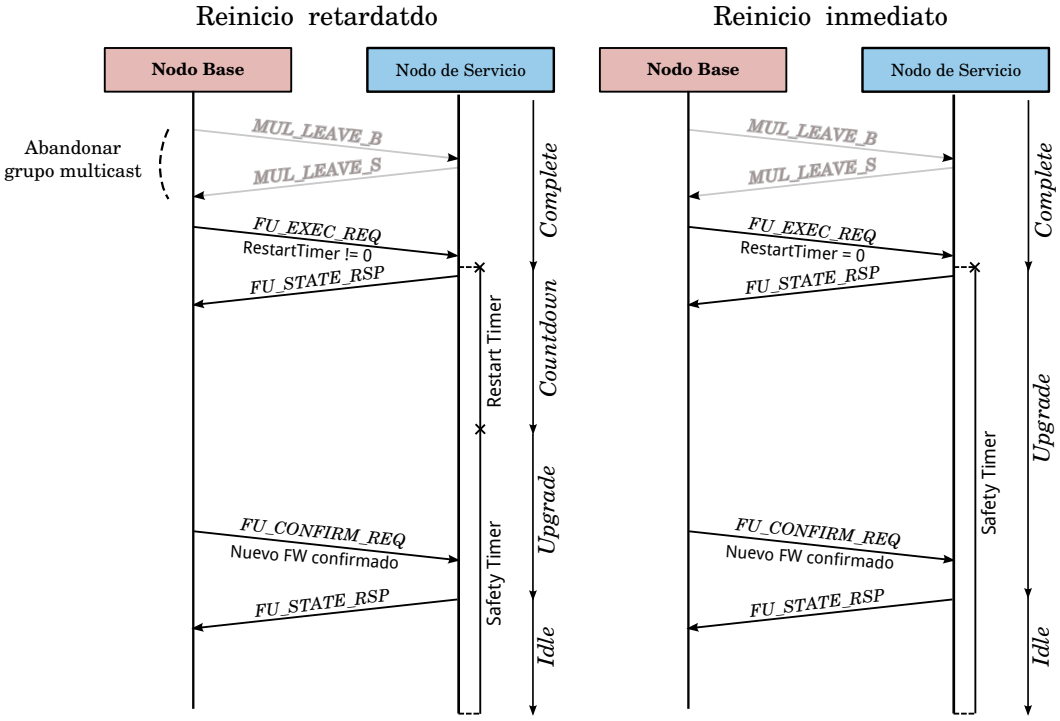


Figura 2.33: Reinicio del nodo de servicio y ejecución del *firmware* (retardado e inmediato)

Capítulo

3

Estado del arte

En este capítulo se incluye una revisión bibliográfica de los modelos de simulación de redes PLC de banda estrecha más importantes existentes en la actualidad. Los distintos modelos de simulación se agrupan por autores y se identifican sus características más importantes.

3.1 Visión general

La contribución principal de este trabajo de investigación es la realización de un estudio que incluye el diseño, implementación y evaluación de distintas estrategias de actualización *firmware* de los equipos de telemedida de una subred PRIME con el objetivo de mejorar la disponibilidad de la subred durante el proceso, sin que suponga un aumento del tiempo necesario para llevar a cabo dicha actualización. Como se comentaba en el capítulo 1, la aplicación de actualización *firmware* es un buen ejemplo de las aplicaciones que más recursos de red demandan y que empeoran de forma significativa la disponibilidad de las subredes PRIME. Dicha aplicación, tal y como se ha visto en el capitulo 2, requiere del reinicio de cada elemento de la red lo que implica un empeoramiento de la disponibilidad de cada uno de ellos, así como la de todos sus elementos dependientes. El problema de la disponibilidad de las subredes PRIME durante el proceso de actualización *firmware* nunca antes ha sido tratado en la literatura. A pesar de que existen numerosas posibilidades a la hora de llevar a cabo la implementación de dicha aplicación, a día de hoy no existe un consenso sobre qué estrategias de actualización *firmware* son las más eficientes, y menos en términos de disponibilidad de red o duración del proceso. Sin embargo, la optimización de este proceso despierta el interés de operadoras eléctricas como Iberdrola.

Para realizar dicho estudio, teniendo en cuenta que realizar estudios exhaustivos de estas características sobre redes reales es impracticable, debido al alto coste en cuanto a tiempo y esfuerzo, es necesario utilizar un modelo de simulación de subredes PRIME. Sin embargo, debido al carácter novedoso de la tecnología PLC, a día de hoy existen pocas herramientas de simulación de redes de este tipo capaces de dar estimaciones de tráfico de red realistas y precisas. Además, a esto hay que añadir que muchas herramientas de este tipo son de carácter privado, como ocurre, por cuestiones estratégicas, con las desarrolladas por los distintos fabricantes de equipos de telemedida. Por esta razón, para la realización de este trabajo de investigación se ha visto necesaria la implementación de una nueva herramienta de simulación de subredes PRIME propia. Para ello, se ha realizado un análisis del estado actual de los modelos de simulación de redes PLC de banda estrecha en la literatura agrupándolos por autores e identificando sus características más importantes y que han sido incluidas en este capítulo.

3.2 Modelos de simulación de redes PLC de banda estrecha

3.2.1 Sanz et al.

Uno de los simuladores de redes PRIME más importantes de la literatura fue presentado en los trabajos [SPMG12, SPMGN12, SPMG13] por miembros de la compañía *ADD Semiconductor* (ahora Atmel). Actualmente una de las áreas de trabajo de Atmel se centra en el desarrollo de circuitos integrados para sistemas sistemas de *Smart Metering*.

En los trabajos citados, se describe un simulador basado en Linux que utiliza algunas de las funcionalidades *Inter Process Communication (IPC)* propias del sistema operativo. La funcionalidad IPC permite que distintos procesos que se ejecutan en paralelo puedan comunicarse y sincronizarse entre sí mediante la compartición espacios de memoria.

El simulador cuenta con un enfoque de eventos distribuido [Mis86] basado en OpenMPI [1][GFB+04], lo que permite simular redes PRIME reales con una buena precision y rendimiento.

La herramienta utiliza el código PRIME que está cargado actualmente en los medidores comerciales reales y emula el funcionamiento de las partes *hardware* principales de un medidor típico consiguiendo así un alto grado de realismo. Esta herramienta ya se ha introducido en el proceso de desarrollo de código de

[1]Se trata de una implementación de código abierto del estándar *Message Passing Interface (MPI)* [GLT99]

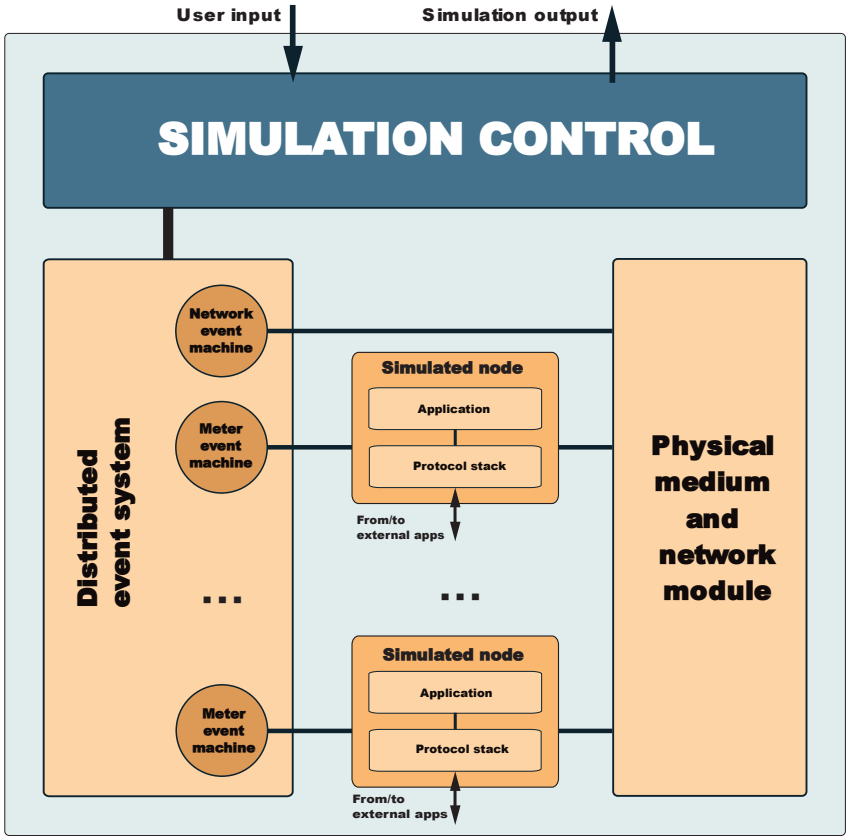


Figura 3.1: Esquema general del entorno de simulación de Sanz et al.

la compañía (Atmel) y se utiliza para mejorar tanto el rendimiento del código como la planificación del despliegue de redes.

El sistema es capaz de distribuir la carga computacional entre varios ordenadores, lo que permite simular redes más grandes en un intervalo de tiempo más pequeño.

Por otro lado, gracias a la flexibilidad de la herramienta, mediante la emulación de la *Universal Asynchronous Receiver-Transceiver (UART)* de los contadores, ésta puede integrarse con otras herramientas de monitorización de redes reales como son el *ADD PRIME Sniffer* o el *ADD Network Manager*. De este modo, es posible supervisar el tráfico en cualquier lugar de la red simulada y comprobar la evolución de la misma.

La estructura del entorno de simulación descrito se muestra en la figura 3.1. Como se puede apreciar, se presenta una estructura orientada a eventos distribuidos, donde cada nodo PRIME en la simulación es un proceso independiente que tiene su propia maquina de eventos. La coherencia de tiempo entre las máquinas de eventos de los nodos se mantiene gracias a una entidad de control de la simulación la cual se encarga de mantener la causalidad de la simulación.

Por otro lado, las comunicaciones entre los nodos se logra a través del módulo de red o *network module*. Dicho módulo de red almacena la estructura de la red y los niveles de ruido en la entrada de los nodos de modo que estos valores puedan ser empleados para determinar los destinos de cada trama transmitida. Finalmente, la herramienta permite lanzar una aplicación conjuntamente con cada nodo y comunicarse con él a través de una UART emulada. Esta aplicación puede ser de cualquier tipo, desde una aplicación de prueba a una aplicación de emulación de contadores. A continuación, se describen las partes más importantes de la herramienta de simulación.

- **Máquina de eventos.** Cada nodo que forma parte de la simulación cuenta con su propia máquina de eventos. Estas máquinas de eventos procesan los eventos a nivel de nodo y los eventos de comunicación entre aplicaciones y nodo. Inicialmente, estas máquinas de eventos están detenidas a la espera de que la entidad de control de la simulación les indique el periodo de tiempo en el que tienen que ponerse en marcha y procesar eventos en paralelo. Al finalizar el periodo de ejecución, envían todo los relativo a los eventos de red generados durante la ejecución (p.ej. transmisión/recepción de tramas) a la entidad de control. Antes de comenzar un nuevo periodo de ejecución, la entidad de control distribuye estos eventos de red entre las demás máquinas de eventos con el fin de mantener la coherencia en la simulación.

- **Módulo de red.** Este modulo es el responsable de las comunicaciones entre los nodos simulados. Recibe todas sus peticiones de transmisión y calcula cuál de ellos recibe correctamente cada trama transmitida. Esto no sólo depende de las características del medio entre cada par emisor-receptor, sino que también de las colisiones que podrían ocurrir durante esta transmisión. A fin de gestionar correctamente todas las comunicaciones, este proceso necesita dos entradas, la estructura de red y el tipo ruido y nivel en la entrada de cada nodo.

  Con el fin de obtener un simulador más realista, se ha empleado una abstracción de la capa PHY similar a la descrita en [KH$^+$10] que ha sido adaptada a distintos protocolos. Mediante esta abstracción, el simulador es capaz de predecir la influencia de distintas condiciones de canal y ruido en las comunicaciones. Los parámetros de este modelo se han obtenido a través de una amplia campaña de medidas en distintos escenarios reales.

- **Módulo de nodo.** En el módulo del nodo simulado se compila el código (PRIME o ECSS) de equipos comerciales reales y se ejecuta como un proceso Linux, obteniendo así un alto grado de realismo. Es importante mencionar que es necesario realizar algunas modificaciones en la capa física y en los drivers del *hardware* para poder adaptar el código real al

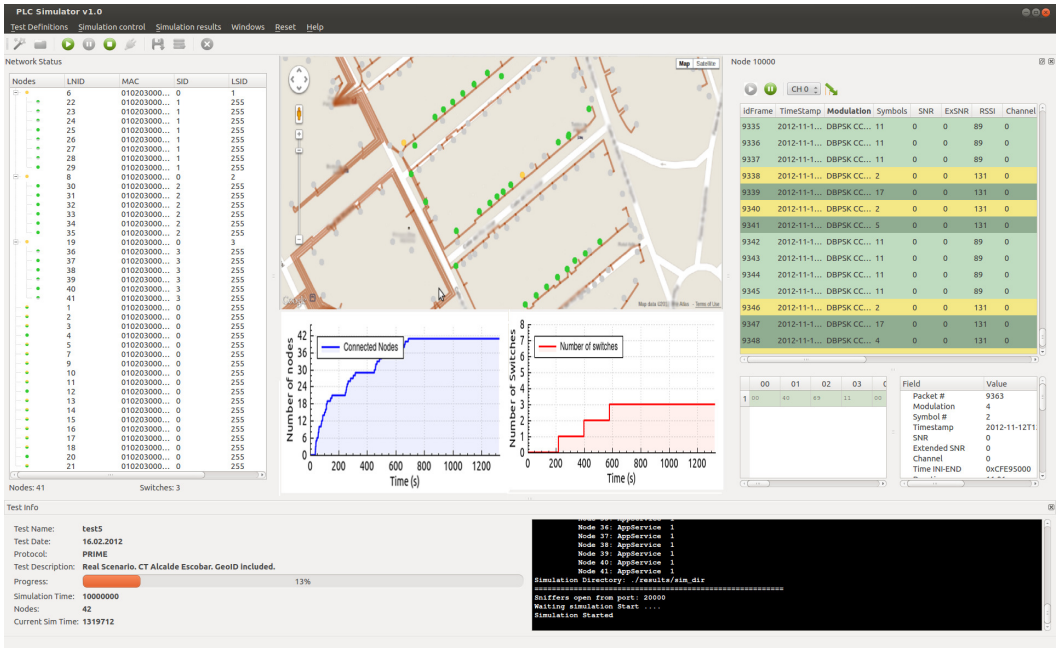


Figura 3.2: Interfaz gráfica de usuario del simulador de Sanz et al.

modelo de simulación. La capa física deberá ser adaptada para transmitir las tramas a través de los procedimientos del modelo de simulación en lugar de emplear el módem PLC. Además, la transmisión y recepción de datos debe hacerse siguiendo el enfoque por eventos. Por otra parte, se deberán emular las partes principales de *hardware* de un nodo real (UART, memoria *flash* o el micro-controlador).

Por otro lado, el simulador cuenta con una interfaz gráfica de usuario o *Graphical User Interface (GUI)* (figura 3.2) cuya funcionalidad puede dividirse en tres partes: descripción del experimento, ejecución de la simulación y análisis de resultados. La descripción de los experimentos se realiza mediante un asistente que traduce la descripción de forma automática a un fichero XML que es la entrada del entorno de simulación. Una vez descrito el experimento, el GUI permite al usuario pausar y reiniciar la simulación como desee. Finalmente, el simulador devuelve una base de datos que puede ser consultada mediante *Structured Query Language (SQL)* que incluye toda la información de la simulación realizada.

Con el objetivo de evaluar el rendimiento y precisión del simulador desarrollado, a lo largo de los tres trabajos citados anteriormente ([SPMG12, SPMGN12, SPMG13]), se han presentado las distintas pruebas realizadas en distintos escenarios y empleando distintos protocolos.

La primera de las pruebas se presenta en [SPMG12], donde con el objetivo de llevar a cabo una primera validación del simulador se ha creado una red PRIME

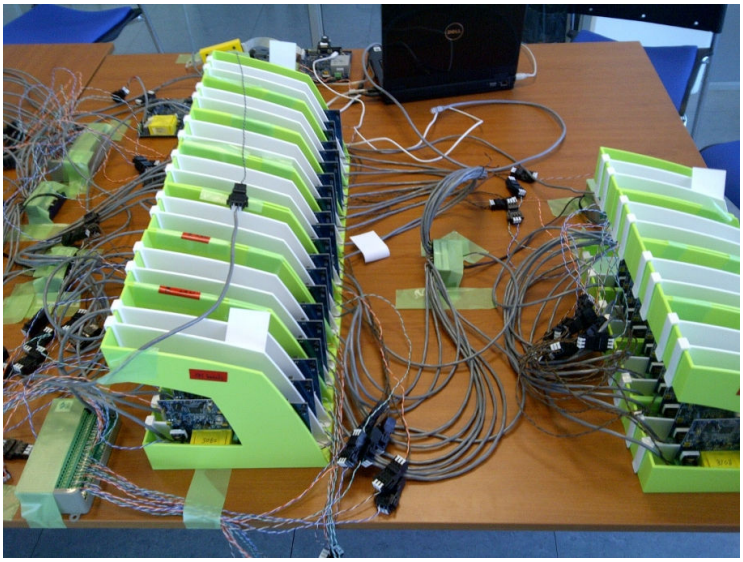


Figura 3.3: Banco de pruebas de laboratorio para la validación del simulador de Sanz et al.

en un laboratorio empleando cien placas de desarrollo *ADDM2101*, tal y como se muestra en la figura 3.3. En dicho proceso de validación se ha introducido la misma estructura de red con el mismo tipo de nodos en el simulador para poder comparar la evolución del proceso de registro de la red tanto en el entorno real como en el simulado.

En el segundo trabajo, [SPMGN12], el simulador descrito se ha utilizado para comparar dos de los protocolos de medición inteligente más importantes en este momento: PRIME y ECSS (ver capítulo 2). En las pruebas realizadas se han empleado dos escenarios. El primero se trata de un escenario de pruebas controlado que cuenta con 30 contadores, conectados en tres grupos y separados por una atenuación de 20 dB. El segundo escenario se ha extraído de un centro de transformación real situado en el centro de una gran ciudad en España. Este escenario se compone de 41 nodos distribuidos a lo largo de tres calles diferentes. Los valores de atenuación y de ruido se han obtenido a partir de mediciones reales.

En ambos escenarios simulados se comparan los protocolos PRIME y ECSS, basándose en tres aspectos importantes:

- **Formación de red:** es el tiempo necesario para establecer comunicación entre el nodo de base y el resto de los nodos en la simulación.
- **Estructura de red:** es el número de niveles de la red.
- **Tiempo de ciclo:** es el tiempo empleado por el concentrador para recoger datos de todos los contadores registrados.

Finalmente, en el trabajo presentado en [SPMG13], el simulador se ha empleado para solventar un problema encontrado en una red real. Por lo general, las solicitudes de datos son más largas que las tramas de control utilizados para el mantenimiento de la red. Este hecho provoca un problema cuando los nodos están conectados a través de un enlace de bajo *Signal-to-Noise Ratio (SNR)*, ya que aunque las tramas de control se pueden recibir correctamente las tramas de datos no. Por esta razón, se fija un valor mínimo de SNR de 3 dB para que se pueda llevar a cabo el registro de los nodos, ya que con este valor de SNR es posible garantizar una probabilidad alta de recepción de datos después del registro de los nodos. Sin embargo, en algunos escenarios, como el expuesto en el artículo, ocurre que el nivel de SNR en los enlaces cercanos a la fuente de ruido es muy bajo, no llegando al mínimo de 3 dB fijado en la mayoría de los casos. Debido a esto, los nodos cercanos a la fuente de ruido se encuentran con grandes problemas a la hora de registrarse. Además, el número de *Switches* creados también es muy alto ya que los nodos que no consiguen registrarse están continuamente enviando tramas de petición de promoción (*PNPDU*) para crear nuevos *Switches* que les permitan registrarse a través de ellos.

El primer paso para solventar el problema es tratar de reproducir el mismo problema en el simulador. Así, se genera un escenario con la misma estructura ajustando los parámetros de atenuación en cada enlace para obtener una probabilidad de pérdida de paquetes similar a la medida en el escenario real.

Para obtener los valores reales de atenuación se han situado *sniffers* en distintos puntos de la red real. Una de las diferencias de la red simulada con respecto a la real es el número de contadores. Las simulaciones con 352 nodos son lentas y no permiten la modificación y testeo ágil del código fuente. Por esta razón, en lugar de simular una red con los 352 nodos solamente se toman en cuenta 160 contadores, por lo que la red simulada es de menor tamaño que la del despliegue real. Los resultados muestran que, a pesar de estas diferencias, el escenario real ha sido modelado correctamente.

3.2.1.1 Análisis

Una de las principales características de la herramienta de simulación presentada en este trabajo es que para simular el comportamiento de los equipos de telemedida, tanto en el caso de PRIME como en el caso de ECSS, utiliza el mismo código que emplean los equipos comerciales de este mismo fabricante (Atmel). Gracias a ello, la herramienta reproducirá fielmente el comportamiento de los nodos de una red real. Además, debido a que los nodos emulan las partes *hardware* como la UART de los nodos, es posible integrarlas con otras herramientas de monitorización de redes reales.

Sin embargo, uno de los inconvenientes de utilizar este método, tal y como destacan los autores, es que el uso del código de los nodos reales supone un alto

consumo de *Central Processing Unit (CPU)*. Este problema es compensado por el uso de varios ordenadores entre los que se distribuye la carga computacional, y/o mediante el uso de una versión simplificada del código de los contadores.

Otro aspecto importante a destacar de la herramienta de simulación descrita es que para determinar si un paquete llega o no a su destino se toman en cuenta aspectos del canal físico como la estructura de la red, incluyendo la atenuación media entre los nodos, y el tipo y nivel de ruido en la entrada de cada nodo. Para obtener un simulador más realista el modelo del canal físico se ha basado en un amplio conjunto de medidas realizadas en diferentes escenarios reales. En concreto, en el trabajo [SPMG13], para reproducir el problema encontrado en la red real descrita en el simulador, se ha creado un escenario con la misma estructura mediante el ajuste de los parámetros de atenuación en cada enlace para obtener una probabilidad de pérdida de tramas similar al medido en el escenario real. Para obtener los valores reales, se han situado *sniffer*s en diferentes puntos de la red. Sin embargo, el principal problema de este método de trabajo es que el tiempo necesario para obtener dichos valores reales es elevado. Por otra parte, a pesar de contar con los parámetros del canal de atenuación y ruido, para que la simulación sea lo más precisa posible, sería necesario conocer también la ubicación física de los equipos de telemedida y conocer así las distancias que hay entre cada par emisor-receptor. Aunque en este trabajo no se hace referencia a ello, esta información no siempre está disponible en los despliegues reales y a pesar de que muchas operadoras afirman tenerla no es 100 % fiable [SBA+13].

Por último, cabe destacar que la principal diferencia de la red simulada con respecto al escenario real es el número de nodos, ya que para poder realizar las pruebas con una mayor agilidad, el número de nodos en el escenario simulado se ha reducido y es de 160 frente a los 352 del entorno real. Por lo tanto, a pesar de los buenos resultados proporcionados por el simulador, la red simulada en este trabajo no es idéntica a la del escenario real.

3.2.2 Mora et al.

Otro de los simuladores de subredes PRIME de la literatura fue desarrollado en el marco del proyecto Gestión Activa de la Demanda (GAD) y se presentó en [MLR+09], el cual fue liderado por Iberdrola y colaboraron empresas como Siemens, CEDETEL y Red Eléctrica de España. El objetivo de este proyecto era diseñar de un sistema de red inteligente con el fin de mejorar el equilibrio entre la generación y la distribución de electricidad en el escenario residencial [gad09].

En el proyecto GAD se define y desarrolla una nueva red de comunicaciones a lo largo de los diferentes segmentos de la red eléctrica (alta, media, baja tensión), modelando una nueva arquitectura de comunicaciones desde las capas

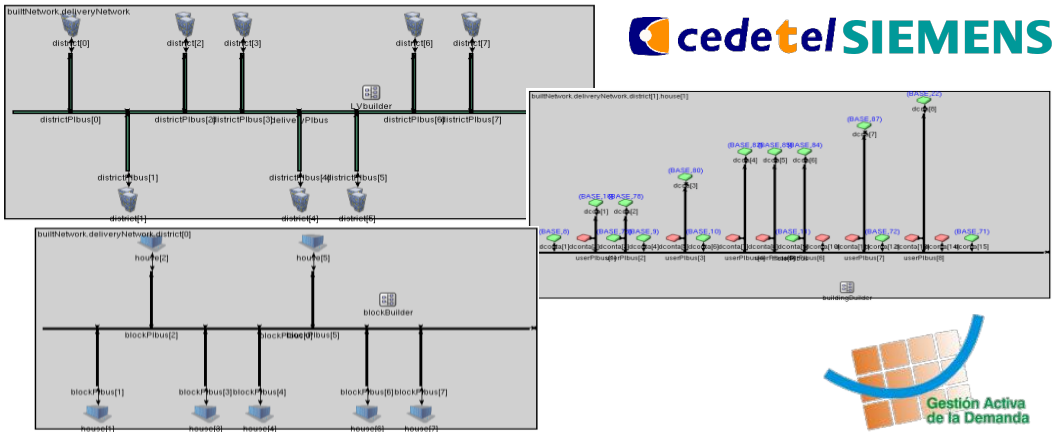


Figura 3.4: Interfaz gráfica del simulador PRIME desarrollado en el proyecto GAD (Mora et al.)

físicas más bajas hasta la de aplicación, así como los protocolos necesarios para las diferentes capas de comunicación. La red de comunicaciones tiene como objetivo garantizar una correcta y segura transmisión/recepción y consolidación de la información hasta la última milla.

Para llevar a cabo el diseño de dicha red de comunicaciones, se ha desarrollado un modelo de simulación basado en OMNeT++ (figura 3.4) que consta de las siguientes partes:

- Un modelo del medio físico que emula las pérdidas de propagación, pérdidas de inserción, las colisiones y el ruido (térmico, de banda estrecha e impulsivo).

- Un modelo de interfaz PHY PRIME basado en fenómenos físicos, que permite la transmisión y recepción de tramas PRIME con control automático de ganancia, detección de cruce por cero, sensibilidad del receptor, diferentes modulaciones (DBPSK, DQPSK y D8PSK), ruido y efectos del decodificador convolucional.

- La capa PHY de PRIME se ha basado en un modelo estadístico para la simulación rápida y masiva de escenarios.

- Una capa PRIME MAC según la especificación (completada en 2009) que soporta la transmisión de balizas, direccionamiento, paquetes múltiples y mecanismo de promoción, entre otras funcionalidades.

- Una capa de convergencia funcional entre IPv4 y la capa MAC de PRIME para permitir las comunicaciones IP en el entorno simulado sobre PRIME.

En relación a este mismo proyecto, el simulador desarrollado se emplea en [MLSB10, MRN+11] para testear nuevos diseños de red antes de que sean desplegados. En concreto, la herramienta se utiliza para emular una región de España de modo que se pueda realizar una primera aproximación de un despliegue nacional y así poder tomar decisiones antes de pasar a un despliegue masivo.

3.2.2.1 Análisis

El modelo de simulación desarrollado está basado en un simulador modular de eventos discretos de redes orientado a objetos llamado OMNeT++. Una de las ventajas de utilizar este simulador es que todas las instancias de los nodos se crean dentro del entorno de simulación, por lo que es la propia herramienta la que se encarga de sincronizar y mantener la causalidad de los eventos de todos los elementos de la red. Además, debido a que OMNeT++ es un simulador de eventos discretos, las simulaciones se ejecutan con mayor rapidez ya que la duración de las simulaciones está ligada al número de eventos que ocurren y a las prestaciones de la máquina en la que se realiza.

Por otra parte, el simulador centra muchos esfuerzos en caracterizar el canal físico, pero el modelado del canal físico es una tarea compleja que no siempre refleja lo que ocurre en la realidad. Además, como ya se ha señalado, aunque el modelado del canal físico fuera preciso, para simular una red real conocida necesitaríamos conocer la ubicación física de todos los nodos, pero esta información no siempre es conocida.

Otro aspecto a destacar es que en este artículo no se incluye ningún proceso de validación de la herramienta a pesar de que para finales del 2009, se esperaba tener una comparativa entre los resultados experimentales con los obtenidos en las simulaciones.

3.2.3 Matanza et al.

El simulador que se describe a continuación, presentado en [MARM13], se basa en la herramienta de software matemático MatLab y en el simulador de redes OMNeT++. La figura 3.5 muestra la arquitectura de la herramienta desarrollada.

Como se observa en dicha figura, los efectos relacionados con la capa física (PHY) de las comunicaciones han sido tomados en cuenta mediante el uso de los resultados de simulaciones de Matlab. Gracias a dichas simulaciones se obtiene el comportamiento del transceptor en términos de *Bit Error Rate (BER)* como una función de SNR. De este modo el BER se usa en el simulador OMNeT++ para computar la probabilidad que tiene un paquete de llegar a su destino con o sin errores. El procedimiento seguido se detalla a continuación.

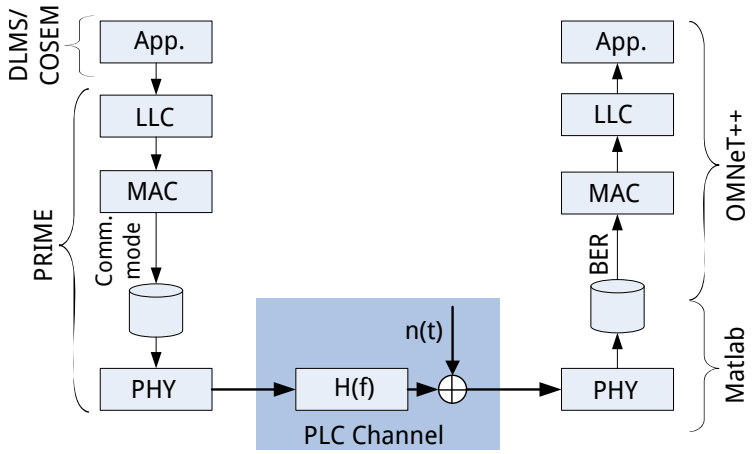


Figura 3.5: Estructura del modelo de simulación de Matanza et al. (basado en OMNeT++ y Matlab)

Cuando se transmite un paquete, éste es atenuado a medida que avanza hasta llegar a todos los nodos de la red. Las atenuaciones para cada par de nodos (transmisor y receptor) son pre-computados por Matlab usando la función de transferencia $H(f)$ del canal PLC.

Tras la recepción del paquete se computa el SNR utilizando el nivel de la señal recibida y el ruido en el escenario actual. Utilizando el SNR calculado y el modo de comunicación de la transmisión se puede extraer el valor BER de las curvas que han sido almacenadas previamente en la memoria de OMNeT++ (figura 3.6). Así, dependiendo del valor del BER y la longitud del paquete para la transmisión, el paquete será correctamente recibido o descartado debido a errores.

La implementación también toma en cuenta los retardos en la propagación entre los nodos. Los retardos se calculan dependiendo de las características físicas de los cables (lo cual permite calcular la velocidad de propagación), la longitud de los mensajes y la distancia entre nodos. Los modelos de cables empleados son el NAYY150E y el NAYY50E.

Por otra parte, se han implementado tanto la capa MAC como la *Logical Link Control (LLC)* (supcapa de convergencia específica de servicio IEC 61334-4-32) definidas en PRIME con OMNeT++. Además, también se ha añadido la capa de aplicación que implementa *Device Language Message Specification/Companion Specification for Energy Metering (DLMS/COSEM)*. Esta parte del modelo simula todos los efectos relacionados con la conexión, la fragmentación y de acceso al canal de acuerdo con los estándares.

Para la simulación del comportamiento del canal PLC el modelo de simulación toma en cuenta dos fenómenos: la función de transferencia del canal y las fuentes de ruido del canal.

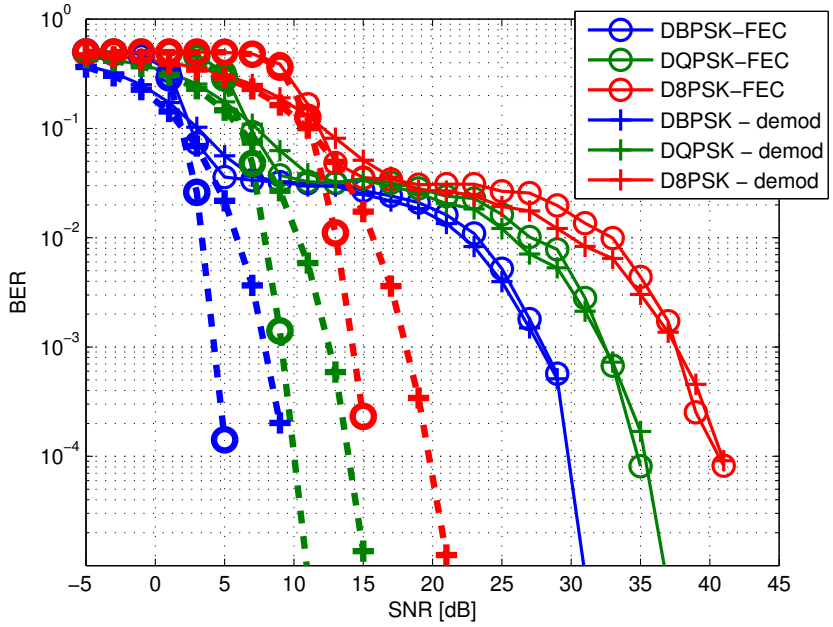


Figura 3.6: Curvas BER-SNR. Rendimiento de PRIME ante la presencia de ruido de fondo (líneas discontinuas) y ruido impulsivo (líneas continuas) (Matanza et al.)

- **Función de transferencia del canal PLC**. La función de transferencia del canal proporciona una relación entre la potencia de la señal de entrada con la de salida en el canal. Esta relación normalmente depende de la frecuencia por lo que algunas frecuencias se verán más atenuadas que otras. Para una topología dada, la función de transferencia de la trayectoria total puede ser calculada mediante el modelado de pequeños segmentos. La matriz general de transmisión es la multiplicación de las matrices de transmisión de todos esos pequeños segmentos. El artículo [DARM11] muestra con más detalle cómo se calculan las funciones de transferencia aplicando la teoría de transmisión de líneas.

- **Ruido del canal**. En el trabajo citado se han tomado en cuenta dos clases de ruidos: ruido de fondo y ruido impulsivo.

  Por un lado, el ruido de fondo está causado principalmente por diversas fuentes de ruido de baja potencia y su densidad espectral de potencia (PSD) se deriva a partir de mediciones, como se muestra en [Hoo98]. Por otro lado, el ruido impulsivo se debe a conmutadores y rectificadores que están conectados a la red eléctrica. En este trabajo se utiliza el modelo Middleton A para emular una fuente de ruido impulsiva, tal y como se ha hecho anteriormente en la literatura [M$^+$10, KHM11].

El simulador descrito se ha empleado para realizar diversos estudios que han sido llevados a cabo por los mismos autores y que se detallan a continuación. Por un lado, en el mismo artículo en el que se describe la herramienta de simulación desarrollada [MARM13], se describe un estudio en el que se analiza el tiempo necesario para realizar la lectura de todos los contadores de una subred PRIME. Para ello, se ha utilizado el modelo de simulación descrito y mediante él se ha emulado una red eléctrica típica europea (incluyendo la función de transferencia y los efectos de ruido). Los resultados muestran que el tiempo requerido depende del número de ramas presentes en la red. Además, este rendimiento ha sido comparado con el valor de referencia de 15 minutos indicado en [NLM+12]. Por otra parte, en este mismo artículo, se ha estudiado la influencia que tiene la posición de un *Switch* en el rendimiento global de la red PLC. Para ello, se ha realizado un experimento en el que se ha analizado el rendimiento de una red PLC compuesta por una sola rama de 200 m con 30 nodos en el que solamente se ha permitido la promoción de un único nodo de servicio. Se han llevado a cabo distintas simulaciones y en cada una de ellas se ha promocionado un nodo de servicio distinto. Después, con cada una de las topologías lógicas se ha realizado la lectura de contadores de manera secuencial.

En el siguiente trabajo [AMRMA14] los autores proponen un nuevo algoritmo de promoción de *Switches* que mejora la latencia en las redes PLC de baja tensión. Para demostrar la efectividad del algoritmo se ha utilizado el modelo de simulación desarrollado en el trabajo presentado en [MARM13]. Las topologías simuladas emulan una red típica europea de baja tensión.

Por último, en [AMRMA15] se analiza el rendimiento de distintas estrategias de lectura simultánea de contadores. En concreto, se estudia el número de conexiones simultáneas que es capaz de manejar el nodo base. Para realizar dicho análisis la métrica utilizada es el tiempo requerido para leer todos los contadores. Para replicar el comportamiento de una red PRIME, y emplearlo así como escenario de pruebas, se ha empleado el simulador descrito en este apartado. Las topologías simuladas en este caso también representan una típica red de baja tensión europea.

3.2.3.1 Análisis

El modelo de simulación presentado en este apartado combina el uso de dos herramientas muy conocidas: Matlab y OMNeT++. Sin embargo, la integración de ambas herramientas no es directa por lo que las simulaciones se realizan en dos pasos. En primer lugar, es necesario realizar simulaciones del canal PLC en Matlab para que los resultados obtenidos en esta simulación (curvas BER-SNR) puedan cargarse en la memoria de OMNeT++ y poder realizar las simulaciones de subredes PRIME.

Un aspecto importante a destacar es que los resultados obtenidos con el modelo de simulación no han sido contrastados con resultados obtenidos en entornos reales o de laboratorio. Por lo que a falta de dicha validación, no podemos asegurar que la herramienta desarrollada sea capaz de predecir el comportamiento de una red PRIME real. Por otra parte, la herramienta centra muchos esfuerzos en la caracterización física del canal PLC y como ya se ha señalado anteriormente, aunque dicha caracterización fuera precisa, si se quisiera reproducir el comportamiento de una red real (p.ej. para validar la herramienta), sería necesario conocer la posición exacta de todos los contadores.

3.2.4 Di Bert et al.

El trabajo presentado en [DBDAT14] propone un simulador multiplataforma que permite simular redes de acceso basadas en G3-PLC. La herramienta consta de dos simuladores: uno para la capa física (PHY) y otra para la capa de enlace de datos (DLL) y de adaptación (ADP). El primer simulador ha sido implementado en Matlab y se utiliza para calcular las respuestas de canal de los enlaces de una red dada. Además, también se calculan dos métricas de rendimiento de la capa PHY: la tasa de error de trama o *Frame Error Rate (FER)* y la tasa de bits de cada enlace. Estos dos indicadores se utilizan para abstraer la capa física (PHY) del simulador de la capa DLL/ADP implementado en OMNeT ++.

El simulador de capa física implementado en Matlab se compone de dos partes: un simulador de topologías de red y un simulador G3-PLC. El simulador de topologías de red genera la respuesta frecuencial del canal entre pares de nodos de red a partir de la descripción física de la red, es decir, a partir de la topología, cables y cargas conectadas. Para ello se emplea el enfoque de abajo-arriba descrito en [TV10]. Este enfoque consiste en realizar un estudio analítico de la red y sus componentes que permite calcular la función de transferencia. La figura 3.7 muestra un ejemplo de una topología de una red de acceso que ha sido empleada en este trabajo. La red de la figura representa un entorno residencial de unos 92.000 m^2 donde hay 25 casas conectadas a 7 ramas que están alimentadas por un único transformador de MT a BT. Cada casa tiene un identificador que va desde el 13 al 58 mientras que el identificador del transformador es 1. Para esta topología se han generado 650 respuestas en frecuencia en la banda de 30-500 kHz. La figura 3.8 muestra algunos ejemplos.

En cuanto al ruido, éste se modela de acuerdo con [TXB+07] y [RKU+11] como un ruido de fondo Gaussiano en la banda de 3-500 kHz. En concreto, se asume que la densidad espectral de potencia del ruido decrece de manera exponencial tal y como muestra la figura 3.9.

El simulador G3-PLC es el encargado de escoger el esquema de transmisión a emplear para el envío de paquetes. Para ello, el estándar define el mecanismo *Adaptive Tone Mapping (ATM)*. En dicho mecanismo, el transmisor le solicita al

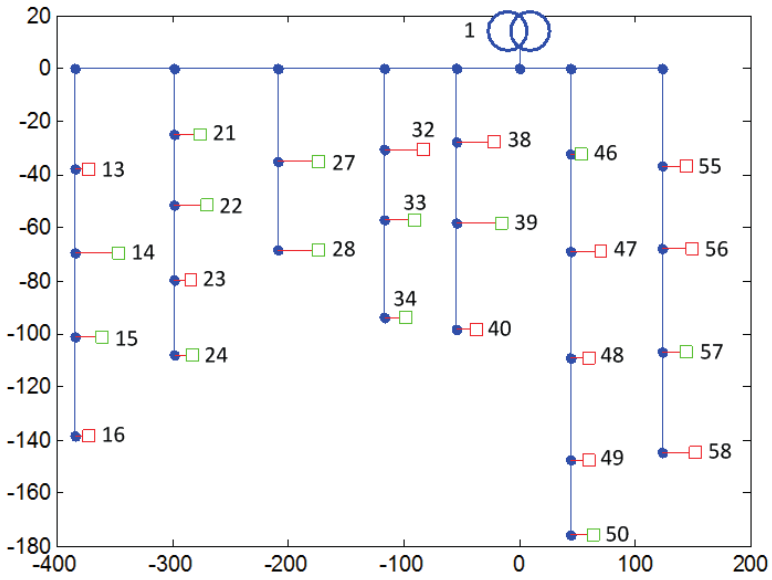


Figura 3.7: Ejemplo de topología de red junto con los IDs de los equipos de telemedida (Di Bert et al.)

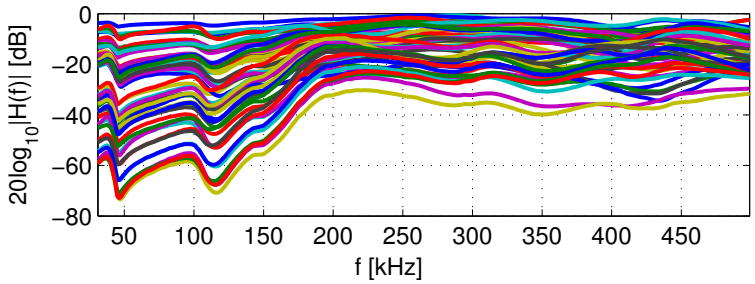


Figura 3.8: Ejemplo de respuestas en frecuencia de la topología mostrada en la figura 3.7 (Di Bert et al.)

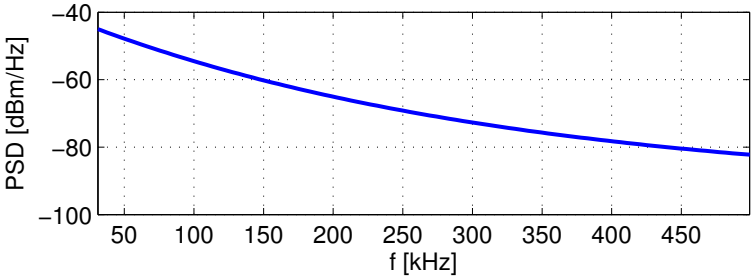


Figura 3.9: Densidad espectral de potencia del ruido de fondo (Di Bert et al.)

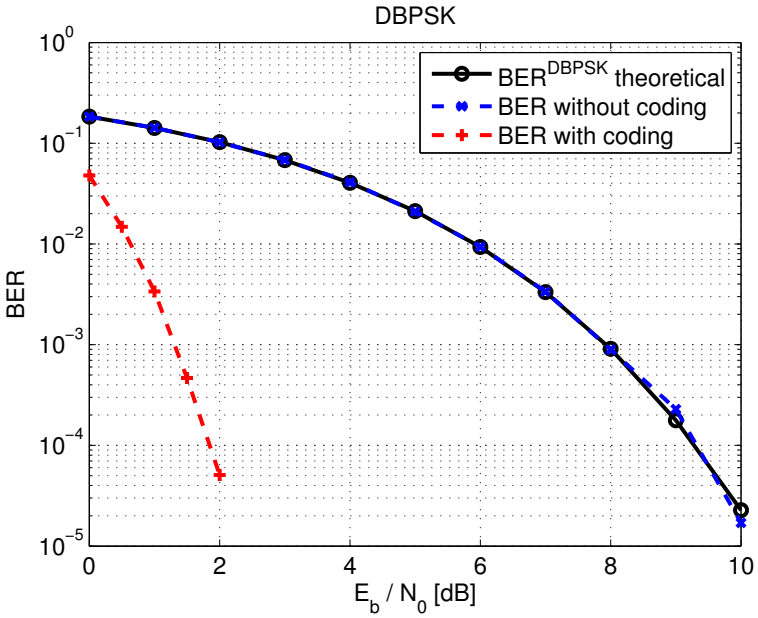


Figura 3.10: BER vs E_b/N_0 (SNR) para la modulación DBPSK con y sin codificación (Di Bert et al.)

receptor que estime la condiciones de canal entre ellos y que escoja los parámetros de capa PHY óptimos para llevar a cabo la comunicación. Por lo tanto, cuando el receptor reciba una petición de este tipo, éste estimará el valor SNR de la señal recibida para cada portadora modulada e informará al transmisor enviándole una respuesta. De este modo, el transmisor será capaz de seleccionar de forma adaptativa las subportadoras disponibles, la modulación a emplear y la tasa de codificación[2] óptima para garantizar una comunicación fiable. En concreto, el transmisor escoge las subportadoras a emplear para el envío de datos y para el envío de símbolos sin información que el receptor deberá descartar. Es importante tener en cuenta que la elección del algoritmo de carga a implementar depende del fabricante del dispositivo. En el trabajo citado, en el algoritmo implementado se fija un valor SNR mínimo de 2 dBs para obtener así un BER de aproximadamente 10^{-4} en el caso de emplear DBPSK con codificación (figura 3.10). Por lo tanto, para el envío de datos se emplearán aquellas subportadoras cuyo valor SNR supere este *offset*. La simulación de la capa PHY descrita está destinada a calcular la tasa de bits y la FER de cada enlace. Estos dos parámetros, a su vez, serán empleados por el simulador de capas superiores (DLL/ADP) implementado en OMNeT++ para abstraerse así de la capa PHY.

Como ya se ha adelantado, el simulador de la capas superiores DLL y ADP se ha implementado mediante el simulador de eventos discretos OMNeT++.

[2]Es la relación entre los bits de información y el numero total de bits enviados.

La simulación de la capa DLL se corresponde con la implementación de la subcapa MAC que está basada en la especificación IEEE 802.15.4–2006 para redes de tipo *Wireless Personal Area Network (WPAN)*. En cuanto a la capa ADP, ésta se basa en 6LoWPAN.

Para verificar el funcionamiento de la herramienta de simulación multiplataforma descrita los autores han realizado varios experimentos. En primer lugar, se ha escogido la red mostrada en la figura 3.7 y se ha realizado una simulación en la que cada nodo le envía 10^4 tramas al coordinador. El objetivo de este experimento es el de evaluar el funcionamiento del mecanismo ATM descrito. Las métricas para evaluar dicho funcionamiento son el BER, FER y el *throughput* medido en kbps con y sin el uso del mecanismo ATM, y sin el uso de nodos intermedios o *Switches*.

En el segundo experimento, cada nodo G3-PLC envía una trama de datos por cada medición realizada al coordinador. Se asume que cada nodo genera tráfico de acuerdo con una distribución exponencial con media igual a 60 s. El experimento se realiza tanto en la banda CENELEC-A como en la *Federal Communications Commission (FCC)* y las métricas empleadas son el *throughput* en bps y el retardo punto a punto en ms.

3.2.4.1 Análisis

La herramienta de simulación presentada en este trabajo tiene como objetivo simular redes de acceso de tipo G3-PLC. Se trata de un simulador multiplataforma que al igual que el presentado en [MARM13] combina el uso de Matlab y OMNeT++. Al igual que en dicho trabajo, las simulaciones en Matlab sirven para reproducir el comportamiento de la capa PHY y así separar dicha capa del simulador de capas superiores (DLL y ADP) implementado en OMNeT++. Es importante mencionar que uno de los inconvenientes de este modelo de simulación es que la integración de estas herramientas no es directa y que, por lo tanto, para reproducir el comportamiento de una red de acceso G3-PLC es necesario realizar las simulaciones en dos pasos: primero en Matlab y después en OMNeT++.

Por otro lado, el simulador de capa PHY se compone de dos partes: un simulador de topologías de red y un simulador G3-PLC. El simulador de topologías de red es el encargado de generar la respuesta frecuencial del canal entre pares de nodos en función de la descripción física de la red. Como ya se ha mencionado, a menudo esta información de la topología física de la red no está disponible por lo que resultaría complicado reproducir el comportamiento de despliegues reales.

Por último, el modelo de simulación presentado no ha sido validado con redes reales o de laboratorio, por lo que no podemos asegurar que los resultados

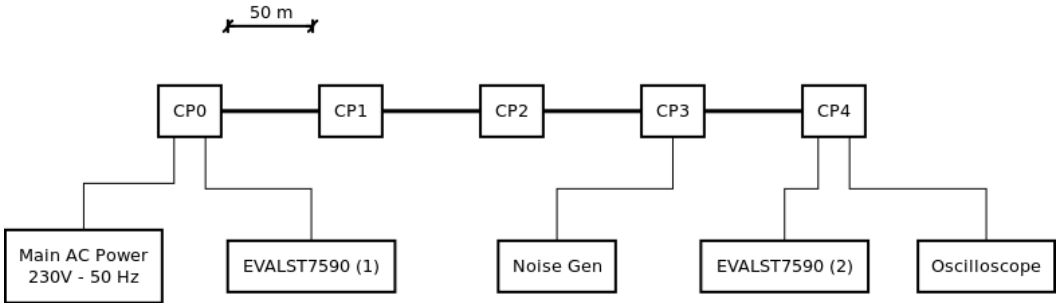


Figura 3.11: Diagrama del banco de pruebas (Patti et al.)

proporcionados por el mismo sean extrapolables a despliegues de redes G3-PLC reales.

3.2.5 Patti et al.

En [PALB13] se presenta un nuevo simulador de subredes PRIME basado en OMNeT++. Tal y como señalan los autores, uno de los problemas que presenta OMNeT++, característico de los simuladores de eventos discretos, es el hecho de que no proporciona funcionalidades específicas para emular los efectos continuos de propagación de la señal, como por ejemplo, la modulación, los modelos de atenuación y el ruido. Para superar este problema el modelado del canal PLC se basa en mediciones realizadas en bancos de pruebas reales. Las mediciones realizadas tienen como objetivo obtener valores BER reales para asociarlos con los distintos valores SNR para todas las modulaciones incluidas en la especificación PRIME. Para realizar dichas mediciones, se ha construido un banco de pruebas compuesto por dos nodos PRIME (EvalST7590), una línea de 200 metros, 5 puntos de conexión separados por 50 metros, un generador de señal (Agilent 33220A) para introducir ruido Gaussiano en el canal y un osciloscopio (figura 3.11).

Las mediciones fueron tomadas en diferentes escenarios de ruido y con diferentes modulaciones, mediante el software PRIME GUI, proporcionado por STMicroelectronics, que permite comunicarse directamente con la capa física desactivando las capas superiores. El SNR se obtiene usando la primitiva *PHY_SNR.get/confirm*, que devuelve un valor de 3 bits que indica el SNR medido del último paquete.

Para obtener el BER, se ha empleado una trama con un *payload* de 377 bytes. Los datos recibidos fueron comparados con los transmitidos para cuantificar el número de bits que sufrieron un error. En cada tanda de medidas, se transmitieron hasta 100 tramas de capa PHY empleando todos los niveles de potencia permitidos bajo distintas condiciones de ruido.

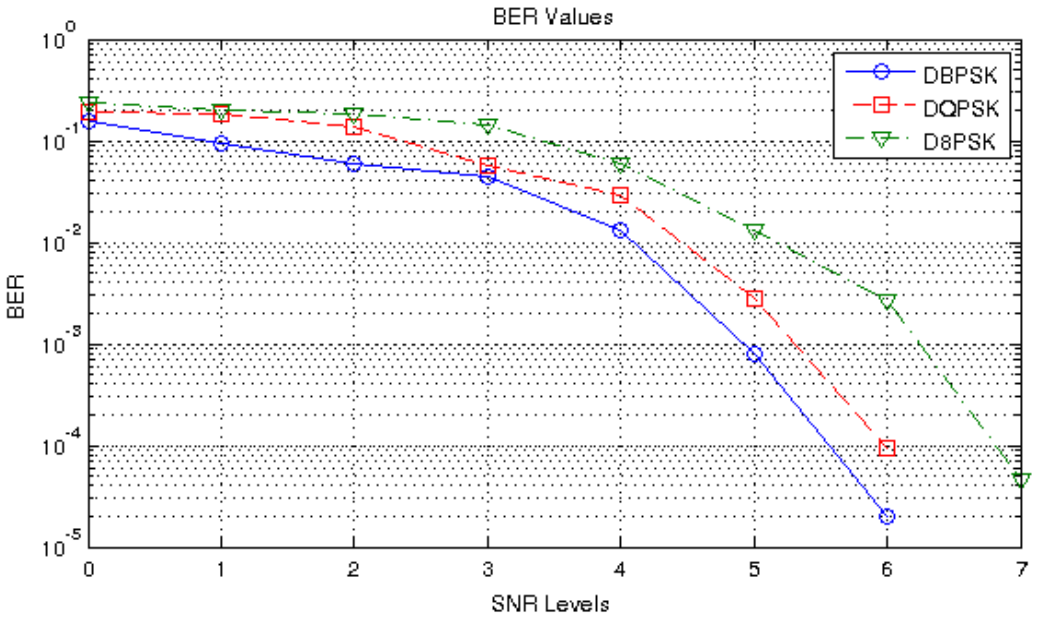


Figura 3.12: Valores BER medidos empleando modulaciones sin FEC activado (Patti et al.)

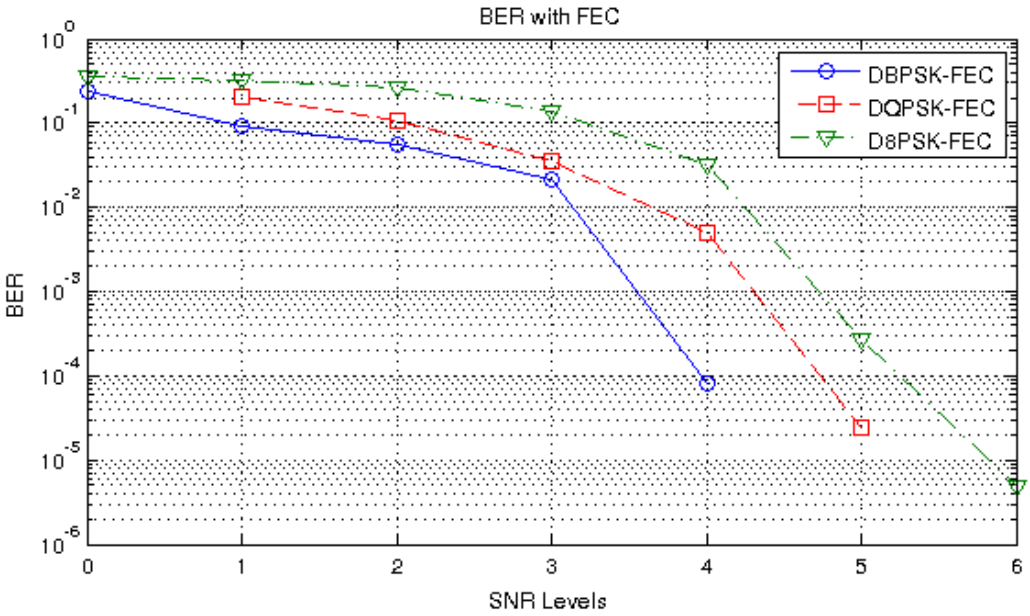


Figura 3.13: Valores BER medidos empleando modulaciones con FEC activado (Patti et al.)

Por otro lado, hay que mencionar que en este trabajo no se implementa un modelo de canal detallado, ya que el objetivo del mismo es solamente evaluar el comportamiento de la capa MAC. Sin embargo, es necesario implementar un modelo de canal aproximado para poder emular las propiedades del canal que tienen impacto sobre la capa MAC, tales como la visibilidad entre nodos y el BER. Por lo tanto, en el entorno de simulación, el canal PLC se simula a través de un módulo que contiene las distancias entre todos los nodos y los valores BER medidos para cada tipo de modulación en el banco de pruebas. La atenuación de la señal se calcula para cada nodo basándose en unos perfiles de longitud propuestos en [ZD02b] aplicando la ecuación 3.1:

$$H(f) = \sum_{n=1}^{N} g_i \cdot e^{-(a_0 + a_1 f^k) d_i} \cdot e^{-j2\pi f \tau_i} \tag{3.1}$$

donde a_0 y a_1 son los parámetros de atenuación y k es el exponente del factor de atenuación. De este modo, cuando un paquete llega, el modulo de canal lee la potencia de transmisión, calcula la atenuación, sustrae el nivel de ruido, y calcula los errores, colisiones y retrasos. Finalmente, el canal envía el paquete a todos los nodos conectados.

Para simular el comportamiento de los nodos se han modelado las capas PHY y MAC siguiendo la especificación de PRIME. La capa MAC se encarga de modelar los siguientes procesos: promoción/des-promoción de los nodos, registro de nodos, establecimiento de conexiones y el mecanismo *Keep-Alive*. Finalmente, se ha implementado la subcapa de convergencia NULL, que realiza un mapeo directo de las primitivas MAC, por lo que el camino desde la capa MAC a capas superiores es transparente.

Con el modelo de simulación desarrollado, en [PALB13], por un lado, se evalúan distintos esquemas de modulación en términos de rendimiento, ratio de entrega de paquetes, retardo punto a punto, etc. Por otro lado, empleando la modulación más robusta, es decir, DBPSK con FEC activado, se analiza el efecto que tienen distintos niveles de carga de tráfico sobre el retardo medio punto a punto en una red de dos niveles. Para realizar los estudios mencionados se emplean distintos escenarios, pero los resultados obtenidos en las simulaciones no han sido contrastados con resultados en campo, de modo que pudiera servir como validación de la herramienta.

3.2.5.1 Análisis

El modelo de simulación presentado en este apartado se centra en analizar el comportamiento de la capa MAC. Sin embargo, implementa un modelo de canal aproximado basado en medidas de un banco de pruebas de un laboratorio.

El modelo de simulación se emplea para realizar distintos análisis de rendimiento de PRIME en distintos escenarios. Sin embargo, los resultados obtenidos

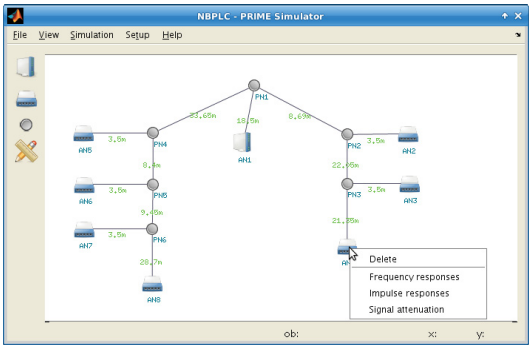


Figura 3.14: Interfaz gráfica de usuario del simulador PrimeSim (Gogic et al.)

en las simulaciones no han sido contrastados con pruebas de campo de modo que pudieran servir de validación del modelo de simulación desarrollado. En caso de querer realizar dicha validación sería necesario conocer la distancia real que hay entre todos los pares de nodos, ya que la atenuación de la señal se calcula en función de la distancia entre nodos (perfiles de longitud). Por lo tanto, no es posible saber si los resultados obtenidos mediante este modelo de simulación sirven para anticipar lo que ocurriría en redes reales.

3.2.6 Gogic et al.

En el trabajo [GMB+14] se presenta un modelo de simulación de capa física de redes PLC de baja tensión de acuerdo con el estándar PRIME. El simulador desarrollado, llamado PrimeSim, ha sido implementado empleando el software matemático Matlab y cuenta con una interfaz gráfica de usuario o GUI para la configuración de red y la gestión de los resultados. En la figura 3.15 se puede observar con más detalle la estructura del simulador PrimeSim.

El GUI del simulador permite configurar la red PLC en cuanto a tamaño y topología (figura 3.14). Esta parte del simulador incluye funciones estándares para guardar y cargar configuraciones de red y topologías, una interfaz para instalar cableado, configurar parámetros de ruido, y funciones tales como la eliminación de nodos y enlaces. El simulador está diseñado de manera que el GUI y el motor del simulador (modelos para el PHY y MAC) puedan ser cambiados y modificados de forma independiente. De este modo, el GUI puede ser utilizado con otros modelos de capa PHY y MAC conformes con otro estándar.

Por otro lado, el simulador permite la inclusión de medidas de ruido, respuesta en frecuencia y probabilidades de error, de tal forma que cada parte del sistema pueda ser analizada por separado y se pueda estimar su rendimiento. La salida del GUI de PrimeSim es un fichero que contiene la matriz de topología de la red, los parámetros del modelo de ruido y los parámetros de línea del cable de transmisión que se usan durante la simulación del modelo PHY. La matriz

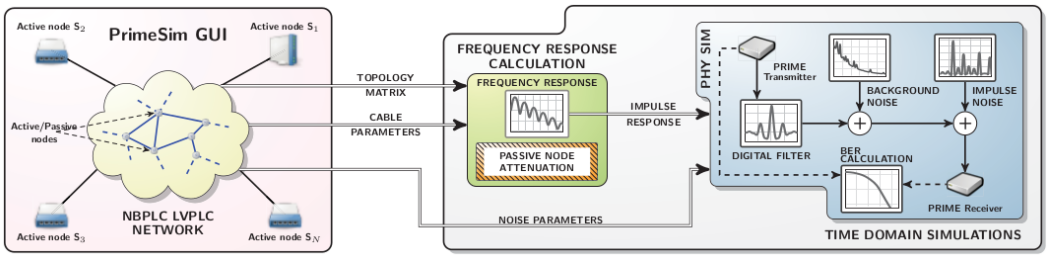


Figura 3.15: Diagrama del modelo de simulación PrimeSim implementado en Matlab (Gogic et al.)

de la topología es una matriz cuyos elementos son objetos de estructura que definen el tipo de nodo (base o nodo de servicio), su ubicación geográfica en la red y su ubicación en la topología lógica.

El modelo del canal de comunicación se basa en el enfoque de abajo-arriba presentado en [Phi98] y en el simulador se ha implementado como un filtro *Finite Impulse Response (FIR)* digital cuya longitud es de 256. El modelo de ruido que incluye tanto ruido de fondo como ruido impulsivo se ha basado en el descrito en [Ben03]. Los parámetros para ambos modelos de ruido pueden ser configurados de forma manual a través del GUI de PrimeSim.

Al finalizar la simulación, los resultados de la misma, es decir, las respuestas en frecuencia y las curvas de probabilidad de error en función del SNR, pueden ser consultados a través del GUI y también pueden ser exportados a ficheros *.csv o *.tr.

La validación del simulador PrimeSim se ha realizado empleando un banco de pruebas que consiste en una red PLC de banda estrecha como la que se muestra en la figura 3.16. Para la construcción de esta red de pruebas se ha empleado cable trenzado de 4 cables donde el área de la sección de cada cable individual es de 16 mm^2. Además, los cables están montados en postes de metal de 4 m de altura.

3.2.6.1 Análisis

Se trata de un modelo de simulación de capa física (PHY) de PRIME que ha sido validado en una red de ocho nodos construida en un laboratorio. Los resultados del simulador se aproximan bastante a los obtenidos en las mediciones y, tal y como apuntan los autores, las diferencias observadas entre los resultados de la simulación con los de la red real son el resultado de las interferencias introducidas por la red de baja tensión principal y que no han sido tomadas en cuenta por PrimeSim. Sin embargo, a pesar de los buenos resultados obtenidos, la herramienta desarrollada no ha sido validada en un despliegue de campo real, donde además las distancias entre nodos no siempre son conocidas.

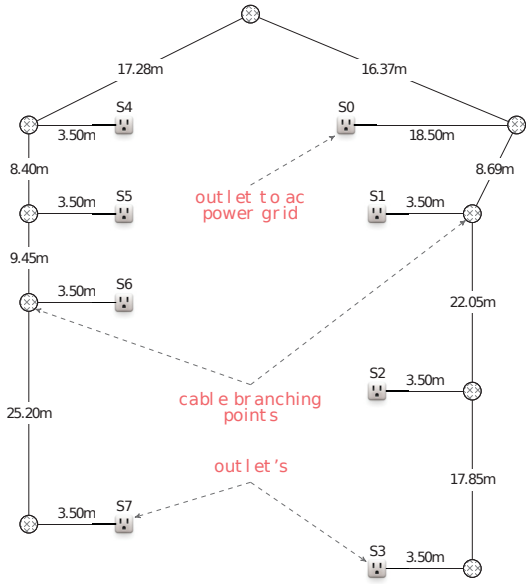


Figura 3.16: Banco de pruebas para la validación de PrimeSim (Gogic et al.)

Por otra parte, es importante mencionar que PrimeSim solamente se centra en la capa PHY de la especificación, por lo que para poder reproducir el comportamiento de redes PRIME reales además de emplear PrimeSim sería necesario contar con una herramienta capaz de simular las capas superiores del estándar.

3.2.7 Gao et al.

En el trabajo presentado en [GYC+08] se describe un modelo de simulación numérico basado en el modelo de Markov de dos estados. Este modelo tiene como objetivo representar la transición de estados de un equipo de telemedida.

En la figura 3.17 los estados "Good" y "Bad" representan dos posibles estados de un equipo de telemedida. El estado "Good" significa que el equipo de telemedida puede comunicarse con el concentrador de datos o nodo base mientras que el estado "Bad" hace referencia al "silent node" que, como se puede deducir de su nombre, no podrá comunicarse con el concentrador. Por otro lado, P_g y P_b son las probabilidades de que un equipo de telemedida se encuentre en un estado "Good" o "Bad", respectivamente, en un entorno de red específico. P_{gg} es la probabilidad de que un equipo que se encuentra en el estado "Good" permanezca en él, mientras que P_{gb} es la probabilidad de que cambie al estado "Bad". Del mismo modo, P_{bb} y P_{bg} son las probabilidades de que un contador en el estado "Bad" permanezca en ese mismo estado o cambie al estado "Good" respectivamente.

Otro parámetro importante a tener en cuenta para modelar el funcionamiento de un equipo de telemedida es el retardo que tiene a la hora de procesar

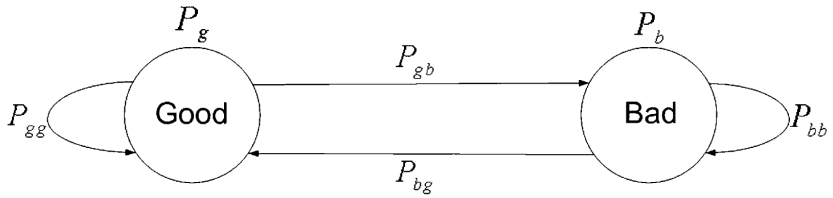


Figura 3.17: Modelo de Markov de transición de dos estados para un equipo de telemedida (Gao et al.)

cada paquete. El valor de este parámetro en este modelo de simulación es de 1 segundo para todos los equipos de la red PLC. Sin embargo, en el caso del concentrador de datos, este valor es despreciable ya que el concentrador es capaz de procesar los datos de las mediciones mientras envía peticiones y recoge las respuestas del siguiente equipo.

El modelo de red de acceso PLC empleado en este trabajo se ha basado en la estructura de edificios de viviendas de protección oficial de Singapur, ya que más del 80 % de viviendas residenciales de este país son de este tipo. La figura 3.18 muestra la estructura de una red de acceso PLC típica de Singapur. Tal y como se observa en dicha figura, la red se divide en tres grupos de equipos de telemedida en función de la distancia que existe desde cada uno de los grupos al nodo base. En concreto, se distinguen tres grupos: "Far", "Medium" y "Near".

De este modo, el tiempo necesario para que el concentrador pueda solicitar y recoger los datos de lectura del contador de cada equipo de telemedida se calcula en función de si se encuentra en el grupo "Far", "Medium" o "Near", y del estado en el que se encuentre ("Good" o "Bad"). Para realizar dicho cálculo también es necesario conocer la tasa de datos de los enlaces y el tamaño de los paquetes a transmitir. Para ello, el trabajo se ha basado en el estándar Konnex (KNX) [kon02] que cuenta con una tasa de $2,4\ kbps$. En cuanto al tamaño de los paquetes a transmitir el escogido es de 24 $Bytes$. Por lo tanto, el tiempo necesario para transmitir un paquete (t_{tr}) en una dirección se calcula mediante la ecuación 3.2.

$$t_{tr} = \frac{\text{Tamaño del paquete}}{\text{Tasa de datos del enlace}} \tag{3.2}$$

El modelo de simulación presentado se ha empleado para analizar el rendimiento de dos mecanismos de lectura automática de contadores o *Automatic Meter Reading (AMR)* conocidos como *Clustered Simple Polling (CSP)* y *Neighbour Rellay Polling (NRP)*.

- **CSP**. En primer lugar, el concentrador solicita las lecturas a todos los equipos de telemedida situados en el *cluster* "Near" uno a uno. Después, el concentrador selecciona uno de los equipos de telemedida del *cluster* "Near"

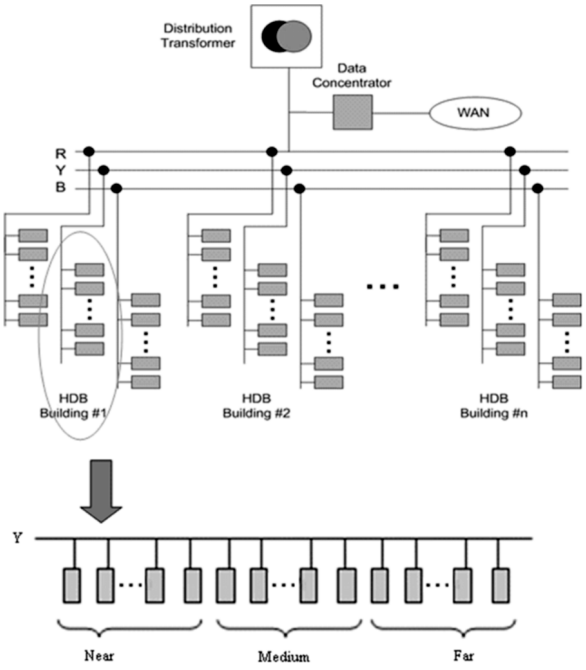


Figura 3.18: Estructura de una red de acceso PLC típica de Singapur (Gao et al.)

que ha respondido correctamente (en estado "Good") para que actúe como retransmisor, de modo que pueda solicitar las lecturas de los equipos del *cluster* "Medium" y éste pueda enviarle los datos al concentrador. Una vez que el concentrador haya recibido los datos solicitados del primer equipo del *cluster* "Medium" o pasado un tiempo no haya recibido nada, le indicará al nodo retransmisor que le solicite los mismos datos a otro equipo de telemedida de dicho *cluster*. El proceso continúa hasta que a todos los equipos del *cluster* "Medium" se les haya solicitado los datos a través del nodo retransmisor. Para el caso de los equipos del *cluster* "Far" el concentrador seleccionará un equipo del *cluster* "Near" y otro del "Medium" para poder comunicarse con dichos equipos. De esta manera, el concentrador enviará un mensaje al primer equipo retransmisor y éste a su vez se lo enviará al segundo equipo rentransmisor para hacérselo llegar a un equipo del *cluster* "Far". El proceso continúa hasta que el concentrador termina con todos los equipos del *cluster* "Far". La figura 3.19 muestra un esquema del funcionamiento de este mecanismo.

- **NRP**. La primera etapa del esquema NRP sigue los pasos del CSP. Después de solicitar los datos de lectura a todos los equipos de medida mediante CSP, si han quedado equipos de los que no se han podido recopilar datos (en estado "Bad" o "silent node") y aún queda tiempo del ciclo de recopilación de datos, el concentrador o retransmisor trata de encontrar un equipo

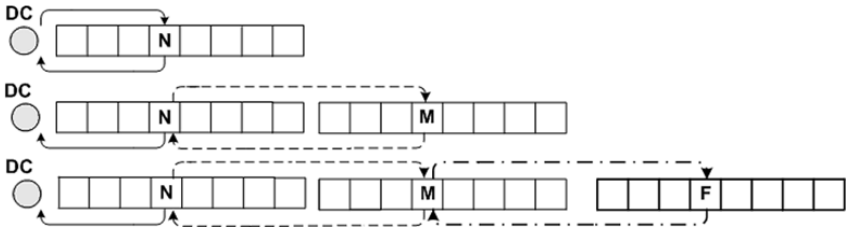


Figura 3.19: Mecanismo CSP de lectura automática de contadores (Gao et al.)

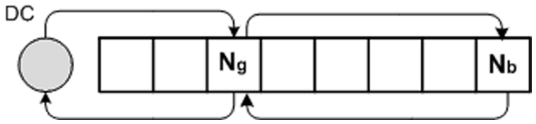


Figura 3.20: Mecanismo de recolección de datos del nodo silecioso N_b en el *cluster* "Near" (mecanismo NRP) (Gao et al.)

de telemedida vecino que se encuentre en estado "Good" (N_g) del nodo en estado "Bad" (N_b) para que actúe como retransmisor (figura 3.20). El concentrador le enviará un paquete al N_g y éste le solicitará los datos de lectura a N_b. Este proceso se repite hasta que se recojan los datos de todos los N_b. Para recoger los datos del *cluster* "Medium", el concentrador envía un mensaje a un equipo con buena comunicación del *cluster* "Near", que a su vez le enviará el mensaje a un N_g del *cluster* "Medium". Finalmente este N_g se encargará de recoger los datos del N_b. Para el caso de los N_b del *cluster* "Far" el proceso es muy similar al descrito.

Por lo tanto, en función del mecanismo de lectura empleado, el *cluster* en el que se encuentre cada equipo y en función de si está en un estado "Good" o "Bad", el tiempo necesario para recoger los datos de lectura del equipo de telemedida será distinto.

En el trabajo citado [GYC+08] después de describir el modelo utilizado, los dos esquemas de lectura automática descritos (CSP y NRP) se comparan utilizando tres métricas: la tasa de éxito de recolección de datos (R_{data}), el tiempo necesario para la recolección de datos de todos los equipos en un ciclo de lectura (T_{round_delay}) y el número de solicitudes de datos adicionales necesarias (N_{add_poll}). En las simulaciones llevadas a cabo se aplicaron diferentes valores de P_{gb} en el modelo de Markov para varios valores de P_g.

Este modelo de simulación también fue empleado en [SGS10], donde se presenta un modelo analítico para estudiar la degradación del rendimiento de los mecanismos CSP y NRP ante la presencia de ruido impulsivo. Para verificar que el modelo analítico utilizado para comparar los dos mecanismos ha sido desarrollado correctamente, los resultados obtenidos se comparan con los del modelo de simulación descrito. Los escenarios empleados para realizar dicho

estudio son de dos tipos: subredes con topología radial (figura 3.21) y de árbol (figura 3.22).

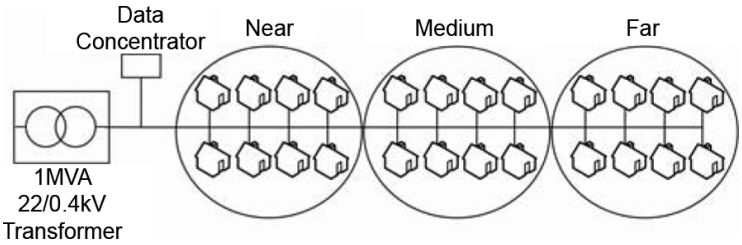


Figura 3.21: Subred con topología radial (Sivaneasan et al.)

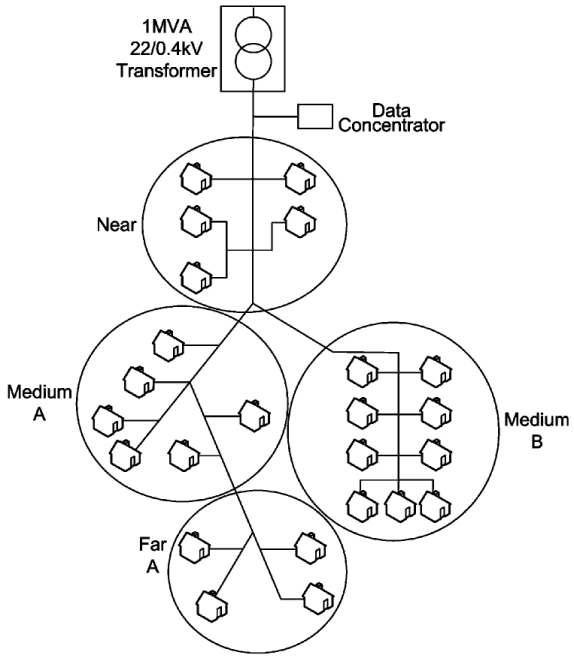


Figura 3.22: Subred con topología de árbol (Sivaneasan et al.)

3.2.7.1 Análisis

El modelo de simulación descrito en este trabajo es muy distinto al presentado en los demás trabajos ya que se trata de un modelo de simulación numérico. Para simular el comportamiento de un nodo se ha implementado un modelo de Markov de dos estados que representa la transición de estados de un equipo de telemedida, el cual puede se puede encontrar en modo “Good” o en modo “Bad”.

Este modelo de simulación se ha empleado para comparar dos mecanismos de lectura automática de contadores (CSP y NRP) en términos de tasa de éxito

de recolección de datos, tiempo necesario para la recolección de datos de todos los equipos en un ciclo de lectura y el número de solicitudes de datos adicionales necesarias. Para calcular estos valores se ha empleado un modelo numérico que variará en función del mecanismo que se desea evaluar. Por lo tanto, el modelo de simulación empleado está muy orientado al problema concreto que se desea solventar.

Los resultados obtenidos no se han contrastado con resultados obtenidos en redes reales, a pesar de que el escenario escogido para realizar las simulaciones es un ejemplo de un tipo de red real de acceso PLC de Singapur. En caso de querer realizar dicha validación, sería necesario conocer la ubicación de los equipos de telemedida de la red real para poder decidir en qué grupo situar a cada uno de ellos (en el grupo“Near”, “Medium” o “Far”).

Finalmente, las simulaciones se realizan asignando distintos valores a las variables P_g y P_{gb}. Sin embargo, para poder predecir el comportamiento que va a tener una red real a la hora de ejecutar el mecanismo CSP o NRP, sería necesario conocer los valores reales P_g y P_{gb} de los equipos de dicha red.

3.3 Resumen del estado actual de modelos de simulación de redes PLC de banda estrecha

En la tabla 3.1 se incluye un resumen de las características más importantes de los modelos de simulación de redes PLC de banda estrecha descritos en este capítulo. La mayor parte de los modelos de simulación se han implementado siguiendo el estándar PRIME, ya que es uno de los estándares más extendidos actualmente. En caso del simulador presentado por Sanz et al., la herramienta también sería válida para reproducir el comportamiento de redes que cumplen el estándar ECSS desarrollado por Atmel. Por otra parte, Di Bert et al. presentan un simulador que reproduce el comportamiento de redes G3-PLC, mientras que el simulador numérico desarrollado por Gao et al. sigue un estándar con menor presencia en Europa llamado Konnex.

En cuanto al tipo de plataforma escogido para el desarrollo de los distintos modelos de simulación también existen algunas diferencias.

Tabla 3.1: Tabla comparativa de simuladores de PLC de banda estrecha existentes en la literatura

Autores	**Estándar**	**Plataforma**	**Capas**	**Ruido**	**Atenuación**	**Validación**
Sanz et al. (2012)	PRIME, ECSS	Procesos Linux (código de contadores comerciales)	PHY MAC CL APP	Modelo descrito en [KH$^+$10] con medidas en campo	Modelo descrito en [KH$^+$10] con medidas en campo	Sí Laboratorio y campo
Mora et al. (2009)	PRIME	OMNeT++	PHY MAC CL APP	Ruido térmico, de banda estrecha e impulsivo (modelo estadístico)	Pérdidas por propagación y por inserción (modelo estadístico)	No
Matanza et al. (2013)	PRIME	Matlab y OMNeT++	PHY MAC CL APP	Ruido de fondo [Hoo98] y ruido impulsivo (modelo Middleton A)	Función de transferencia por segmentos de red [DARM11]	No
Di Bert et al. (2014)	G3	Matlab y OMNeT++	PHY MAC	Ruido de fondo Gaussiano [TXB$^+$07] [RKU$^+$11]	Función de transferencia [TV10]	No
Patti et al. (2013)	PRIME	OMNeT++	PHY MAC	Ruido Gaussiano (banco de pruebas)	Función de transferencia (perfiles de longitud) [ZD02b]	No
Gogic et al. (2014)	PRIME	Matlab	PHY	Ruido de fondo y ruido impulsivo [Ben03]	Función de transferencia [Phi98]	Sí Laboratorio
Gao et al. (2008)	Konnex	Simulaciones numéricas (modelo de Markov)	APP	-	-	No

En el caso del presentado por Sanz et al., para simular el comportamiento de un nodo, se emplea el código de contadores comerciales reales (PRIME o ECSS) y se ejecuta como un proceso Linux, por lo que este simulador ofrece un alto grado de realismo. Por otra parte, los modelos de simulación presentados por Matanza et al. y Di Bert et al. son de tipo multiplataforma y ambos emplean la herramienta Matlab para simular la capa PHY y OMNeT++ para simular las capas superiores de PRIME y G3 respectivamente. En cambio, los simuladores presentados por Mora et al. y Patti et al., debido a que el modelo de la capa PHY implementado es menos complejo que en los dos anteriores, solamente utilizan OMNeT++ para simular tanto la capa PHY como MAC (y superiores en caso de Mora et al.). En cuanto a la herramienta presentada por Gogic et al., llamada PrimeSim, se trata de un simulador de capa PHY de PRIME basado en Matlab que cuenta con un entorno gráfico. Por último, en Gao et al. se presenta un simulador numérico de nivel de aplicación, destinado a evaluar el rendimiento de dos mecanismos de lectura automática de contadores conocidos como CSP y NRP.

Es importante destacar que la mayor parte de los modelos de simulación centran muchos esfuerzos en la caracterización física del canal PLC. Para simular los fenómenos físicos del canal, como el ruido y la atenuación de la señal, los distintos autores se han basado en distintos modelos (existentes en la literatura) y formas de proceder. Por ejemplo, en Sanz et al. se emplea la abstracción de la capa PHY descrita en [KH$^+$10]. Los parámetros del modelo se han obtenido a través de distintas medidas tomadas en escenarios reales. Por otro lado, el simulador desarrollado Mora et al. incluye un modelo del medio físico que emula las pérdidas de propagación, pérdidas de inserción, colisiones y el ruido (térmico, de banda estrecha e impulsivo). Dicha capa física se ha basado en un modelo estadístico para la simulación rápida y masiva de escenarios. Sin embargo, no se incluyen los detalles del modelo estadístico empleado. En el modelo de canal PLC del simulador descrito por Matanza et al., se toman en cuenta dos tipos de ruido: ruido de fondo y ruido impulsivo. Para caracterizar el ruido de fondo el modelo se ha basado en el trabajo presentado en [Hoo98] mientras que para el ruido impulsivo se ha empleado el modelo Middleton A. Por otra parte, para calcular la función de transferencia del canal PLC, éste se ha realizado mediante el modelado de pequeños segmentos de red siguiendo el método descrito en [DARM11]. En el simulador de Di Bert et al., el ruido se modela de acuerdo con [TXB$^+$07] y [RKU$^+$11] como un ruido Gaussiano cuya densidad espectral de potencia decrece exponencialmente con la frecuencia. Por otro lado, para calcular la atenuación de la señal, se genera la respuesta frecuencial entre pares de nodos empleando el enfoque de abajo-arriba descrito en [TV10]. En el caso del simulador descrito por Patti et al., el modelado del canal PLC se basa en mediciones realizadas en un banco de pruebas construido en un laboratorio. Estas mediciones tienen como objetivo obtener valores BER reales para asociarlos a

distintos valores SNR para todos los tipos de modulaciones incluidas en el estándar PRIME. En dichas pruebas se emplea un generador de señal para introducir un ruido Gaussiano en el canal. La atenuación de la señal se calcula mediante un modelo de canal simplificado basado en perfiles de longitud descrito en [ZD02b] y aplicando la ecuación de la función de transferencia correspondiente. Finalmente, en el modelo de simulación de capa física presentado por Gao et al. el modelo de ruido se basa en [Ben03] e incluye tanto el ruido de fondo como el impulsivo. En cuanto al modelo de canal desarrollado, mediante el cual se calcula la atenuación de la señal, este se basa en el enfoque de abajo-arriba descrito en [Phi98].

Por último, las únicas herramientas que han sido validadas son la presentada por Sanz et al. y el modelo de simulación de capa PHY de Gogic et al. El simulador de Sanz et al. se ha validado tanto en una red construida en un laboratorio como en una red real. Sin embargo, el método empleado para simular la red real es algo costoso, ya que para obtener la misma estructura de red ajustando los parámetros de atenuación en cada enlace y así obtener una pérdida de paquetes similar al medido en el escenario real, es necesario situar *sniffers* en distintos puntos de la red real. Por otra parte, no hay que olvidar que la red real y la simulada no son idénticas ya que para agilizar la simulación en lugar de simular una red de 352 nodos solamente se toman en cuenta 160. En el caso del simulador de capa física presentado por Gogic et al. en cambio, la validación se lleva a cabo mediante pruebas realizadas en una red de ocho nodos construida en un laboratorio.

El modelo de simulación de subredes PRIME propuesto en este trabajo de investigación emplea la topología lógica más común de la red proporcionada por el nodo base para reproducir el comportamiento de redes reales. De este modo, no es necesario conocer la ubicación exacta de los nodos en la red como ocurre en la mayoría de los modelos de simulación analizados, ya que además en la mayoría de los despliegues reales no suele ser conocida. Por otra parte, para determinar los destinos de cada trama transmitida no es necesario conocer las atenuaciones para cada par de nodos (transmisor y receptor), por lo que no es necesario colocar *sniffers* en la red real. El capítulo 4 se proporciona más información acerca del modelo de simulación desarrollado y la validación mediante resultados de pruebas reales que se ha llevado a cabo.

Capítulo

4

Modelo de simulación de subredes PRIME

En el presente capítulo se realizará una descripción del modelo de simulación de subredes PRIME desarrollado, que ha sido empleado en el estudio de estrategias de actualización *firmware* descrito en el capitulo 5. El capítulo se encuentra estructurado de la siguiente manera: la sección 4.1 incluye una descripción y un análisis comparativo de los simuladores de redes existentes actualmente, la sección 4.2 incluye una descripción detallada del modelo de simulación de subredes PRIME desarrollado, y por último, la sección 4.3 describe la validación de dicho modelo de simulación.

4.1 Simuladores de redes

En esta sección se presenta una visión general de las herramientas de simulación de redes más importantes que existen en la actualidad.

4.1.1 Network Simulator (NS)

El simulador de redes *ns-2* [ns209] es actualmente el más utilizado en los círculos académicos y de investigación. Se trata de un simulador de código abierto que se desarrolla de forma colaborativa. Este simulador emplea dos lenguajes: los modelos de simulación son *scripts* escritos en *Tool Command Language (Tcl)* mientras que el *kernel* de simulación y los componentes (protocolos, canales, agentes, etc.) se implementan en C ++ y son accesibles a través del lenguaje Tcl. Por otro lado, los escenarios de simulación (topologías de red) se expresan como parte de la secuencia de comandos Tcl, que a su vez también se emplea en

el establecimiento de parámetros para la configuración del comportamiento de las aplicaciones y el registro de las estadísticas. Es importante mencionar que es posible generar un *script* Tcl a partir de una representación gráfica, pero no a la inversa: los editores gráficos no son capaces de entender los *script*s de *ns-2* ni permiten que el usuario los edite gráficamente.

Actualmente, existe una nueva versión del simulador conocida como *ns-3* [ns311] que incluye un rediseño completo de la herramienta para facilitar su uso y mantenimiento así como la inclusión de futuras extensiones. Al igual que su predecesor, *ns-3* se basa en C ++. Sin embargo, *ns-3* ya no usa *scripts* Tcl para controlar la simulación, abandonando así los problemas que se introdujeron por la combinación de C ++ y Tcl. En lugar de ello, las simulaciones se pueden definir mediante programas en C++ o a través de *scripts* en Python.

4.1.2 OMNeT ++

OMNeT ++ [omn12] es un simulador de eventos discretos basado en C++ para el modelado de redes de comunicaciones, multiprocesadores y otros sistemas distribuidos o paralelos. OMNeT++ es de código abierto y puede ser utilizado bajo la licencia *Academic Public License* lo que hace que el *software* sea gratuito para su uso sin ánimo de lucro. También existe una versión para su uso comercial denominada OMNEST [omn14].

Los modelos de simulación en OMNeT++ se componen de módulos que se comunican a través de mensajes. Estos módulos pueden ser simples o compuestos. Los módulos simples se implementan en C++ y son los elementos más pequeños de la jerarquía de módulos por lo que no pueden ser divididos. Los módulos simples pueden ser agrupados en módulos compuestos. Por ejemplo, múltiples módulos simples que implementan distintos protocolos se pueden combinar en un módulo compuesto que representa a un *host*. La composición de estos sencillos módulos en módulos compuestos y por lo tanto la puesta a punto de la simulación de la red se lleva a cabo mediante el lenguaje *Network Description (NED)*. El lenguaje NED se convierte en código C ++ de forma transparente cuando el modelo de simulación es compilado en su conjunto.

4.1.3 Riverbed (OPNET) modeler

Riverbed modeler (anteriormente conocido como OPNET) [opn14] es un simulador comercial de amplio uso en el ámbito empresarial y educacional que además cuenta con una edición limitada para uso no comercial. OPNET tiene una de las selecciones más grandes de protocolos ya implementados listos para su uso como: IPv6, MIPv6, WiMAX, QoS, Ethernet, MPLS, OSPFv3, etc.

Se trata de un simulador de eventos discretos basado en C++. Una de las ventajas más importantes de esta herramienta es que integra los procesos de

modelado, simulación y análisis de redes. Además, a diferencia de las herramientas anteriores, todo el proceso se realiza de forma gráfica, lo que simplifica enormemente su uso y aprendizaje.

4.1.4 Elección del simulador de redes

La tabla 4.1 resume y compara algunas de las características de los simuladores de redes descritos. Después de observar las diferencias entre los distintos simuladores, se ha optado por trabajar con OMNeT++.

Una de las ventajas de utilizar OMNeT++ frente a *ns-2*, que es la herramienta de simulación de redes más extendida, es que *ns-2* no sigue la misma clara separación entre el *kernel* de simulación y los modelos. El simulador *ns-2* contiene los modelos junto con su infraestructura de apoyo como una unidad inseparable. Esta es una diferencia clave ya que mientras que *ns-2* es un simulador de redes en sí mismo, OMNeT++ es una plataforma de simulación en la que el usuario puede construir sus propios modelos de simulación. Por otro lado, *ns-2* carece de muchas herramientas y componentes de infraestructura que OMNeT++ proporciona como por ejemplo, la capacidad de soportar modelos jerárquicos, un editor gráfico, entorno de ejecución basado en GUI, separación de los modelos de los experimentos, herramientas de análisis gráfico, recogida resultados, simulación en paralelo integrada, etc [VH08].

Como se ha comentado, *ns-3* es una versión mejorada de su predecesor *ns-2* en la que una de las diferencias más importantes es que en lugar de utilizar *scripts* Tcl para definir las simulaciones, esta tarea se realiza mediante programas en C++ o a través de *scripts* en Python.

La tarea de definir escenarios de simulación en OMNeT++ escenarios es más sencilla que en *ns-2* y *ns-3* ya que es posible hacerlo mediante un lenguaje intuitivo llamado NED o mediante la interfaz gráfica. OPNET también cuenta con la posibilidad de crear topologías mediante su editor gráfico, pero en este caso, los modelos de OPNET son siempre de topología fija mientras que en OMNeT++ el lenguaje NED y el editor gráfico permiten definir topologías parametrizables. En el caso de OPNET, el editor almacena los modelos en un formato binario propietario, lo que significa que en la práctica es difícil generar modelos de OPNET desde programa. Para ello, sería necesario escribir un programa en C que usa una API de OPNET, mientras que en OMNeT++ los modelos son simples ficheros de texto que pueden ser generados por ejemplo mediante el lenguaje de programación Perl.

Por otro lado, tanto OPNET como OMNeT++ proporcionan un depurador gráfico con animación automática que es esencial a la hora de desarrollar los modelos de forma sencilla. Los simuladores *ns-2* y *ns-3*, en cambio, no contienen una herramienta gráfica por defecto, por lo que es muy común el uso de herramientas independientes como NetAnim o PyViz para proporcionar anima-

Tabla 4.1: Comparativa de simuladores de redes

	ns-2	**ns-3**	**OPNET**	**OMNeT++**
Lenguaje	C++	C++	C++	C++
Escenarios	Tcl	C++ Python	Interfaz gráfica	NED Interfaz gráfica
Animación integrada	No	No	Sí	Sí
Código abierto	Sí	Sí	No	Sí
Licencia	GNU GPLv2	GNU GPLv2	Edición limitada	Uso académico

ción a las simulaciones. NetAnim utiliza el fichero de trazas generado durante las simulaciones para visualizar la topología y animar el flujo de paquetes entre los nodos. Por lo tanto, la animación se lleva a cabo una vez que las simulaciones ya han terminado. PyViz en cambio, diseñado para *ns-3*, es un visualizador de simulación en vivo, lo que significa que no utiliza ningún archivo de trazas. El hecho de emplear herramientas de animación independientes puede suponer que surjan algunos problemas de integración como por ejemplo podrían ocurrir si los ficheros de trazas no se han generado en el formato correcto durante la simulación, teniendo que repetir la simulación.

Otro aspecto a tener en cuenta, es que OPNET, al ser una herramienta comercial (a diferencia de OMNeT++, *ns-2* y *ns-3*), no proporciona el código fuente del *kernel* de simulación (a pesar de que viene con el código fuente de los modelos de los protocolos), por lo que no permite la depuración a nivel de código fuente.

Finalmente, otro motivo importante por el cual se ha escogido OMNeT++ para el desarrollo del modelo de simulación de subredes PRIME presentado en este trabajo de investigación, es el hecho de que en el capítulo 3 se ha observado que numerosos autores han optado por utilizar esta misma herramienta para desarrollar sus modelos de simulación, por lo que se deduce que se trata de la herramienta que mejor se ajusta a las necesidades de este trabajo.

4.2 Modelo de simulación de subredes PRIME

Tal y como se ha expuesto en capítulos anteriores, para llevar a cabo este trabajo de investigación se ha visto necesario el desarrollo de un modelo de simulación de subredes PRIME propio. En los siguientes apartados, se incluye una descripción detallada de las partes más importantes de dicho modelo.

4.2.1 Arquitectura del modelo de simulación

El modelo de simulación de subredes PRIME desarrollado se compone de los siguientes elementos: el nodo base, los nodos de servicio, las líneas de alimentación (*LV feeders*) y un módulo llamado *"Network Manager"* o gestor de red. Este último es un módulo que conoce el estado de toda la red, incluyendo el estado de los nodos de servicio y las líneas. Además, permite saber si dos nodos pueden escucharse entre sí y si se ha producido una colisión.

Los nodos del modelo, es decir, tanto el nodo base como los nodos de servicio, aprovechando que OMNeT++ permite crear módulos compuestos y siguiendo el estándar PRIME, tienen una estructura que se compone de las siguientes capas: capa física (PHY), capa MAC, capa de convergencia (CL) y capa de aplicación (APP). La estructura de los nodos se describirá en detalle más adelante en el apartado 4.2.5 de este capítulo. Por otro lado, la figura 4.1 muestra la apariencia del modelo de simulación.

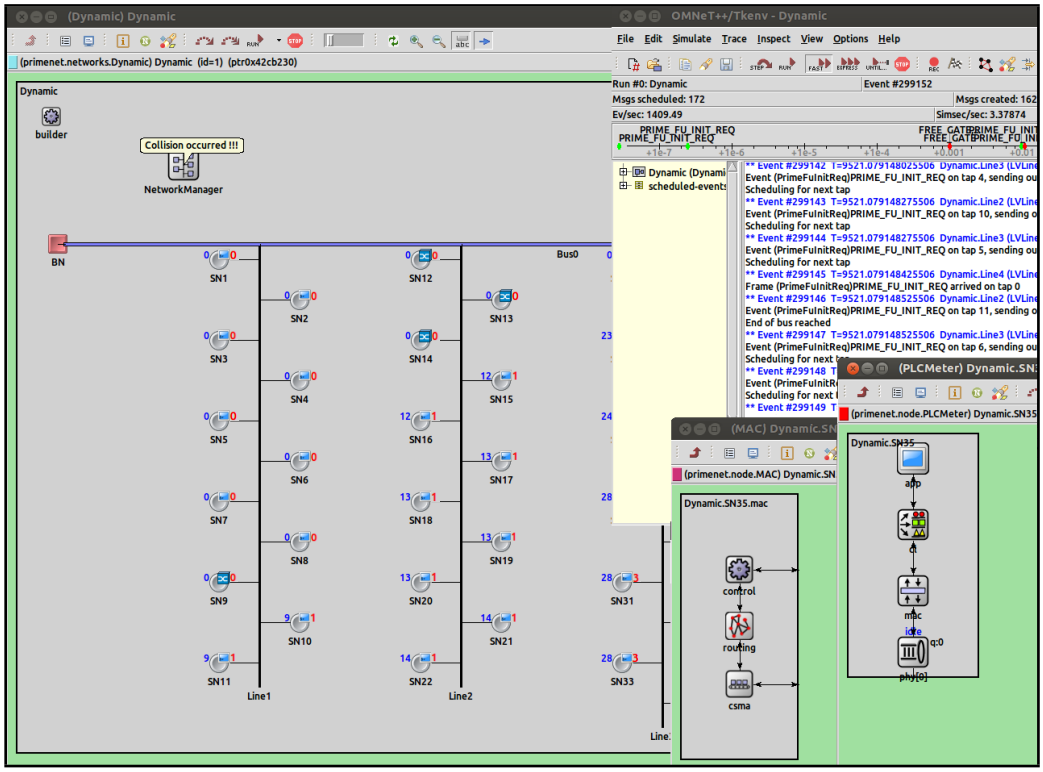


Figura 4.1: Apariencia del modelo de simulación de subredes PRIME

4.2.2 Constructor de la red

OMNeT++ permite realizar la descripción de los escenarios del modelo de simulación de dos modos: de forma gráfica o mediante el lenguaje NED. En

el modelo desarrollado, para facilitar la definición masiva de escenarios se ha optado por utilizar el lenguaje NED.

Existen dos formas principales de crear una red. Por un lado, dentro de un fichero de tipo NED, se crean las instancias de los módulos (simples o compuestos) que componen la red y las conexiones que hay entre ellos. En el ejemplo mostrado en la figura 4.2 se puede observar la forma en la que se define una red empleando el lenguaje NED. La apariencia de la red creada de este modo se muestra en la figura 4.3. Como se puede observar, la red tiene una topología fija que consta de tres nodos y dos canales de comunicación. Esta forma de crear redes tiene un inconveniente, y es que si se quieren realizar simulaciones sobre distintos escenarios, sería necesaria la generación de un fichero NED por cada uno de ellos y compilarlo junto con el resto de las clases que componen el modelo.

```
simple Host
{
    parameters:
        @display("i=block/routing");
    gates:
        input in[];
        output out[];
}
network Network
{
     submodules:
        host0: Host {
            @display("p=61,50");
        }
        host1: Host {
            @display("p=139,94");
        }
        host2: Host {
            @display("p=61,141");
        }
    connections:
        host0.out++ --> { delay = 100ms; } --> host1.in++;
        host0.in++ <--  { delay = 100ms; } <-- host1.out++;

        host1.in++ <--  { delay = 100ms; } <-- host2.out++;
        host1.out++ --> { delay = 100ms; } --> host2.in++;
}
```

Figura 4.2: Ejemplo de un fichero NED que define una red compuesta por módulos de tipo Host

Como alternativa a esta solución, es posible crear redes a partir de ficheros de texto. Para ello, siguiendo el ejemplo de la figura 4.4 se crea un fichero de tipo NED en el que se define un módulo simple llamado *"NetBuilder"* y uno compuesto de tipo *network* llamado *"Dynamic"* que a su vez está compuesto por un módulo de tipo *"NetBuilder"*.

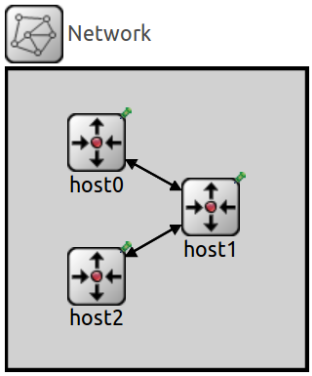


Figura 4.3: Apariencia de red creada a partir del fichero NED de ejemplo

```
simple NetBuilder
{
    parameters:
        @display("i=block/cogwheel_s");
        string nodesFile;
        string connectionsFile;
        string linesFile;
        string busbarsFile;
        string lineConnections;
        string busbarConnections;
}
network Dynamic
{
    submodules:
        builder: NetBuilder {
            nodesFile;
            linesFile ;
            busbarsFile;
            connectionsFile;
            lineConnections;
            busbarConnections;
        }
}
```

Figura 4.4: Ejemplo de fichero NED para la creación de redes desde ficheros de texto

El módulo simple *"NetBuilder"* incluye parámetros de tipo *string* a los que se les asignará la ruta de los ficheros de texto que contienen la información de la red que se desea crear. La asignación de la ruta de estos ficheros de texto se puede realizar directamente en el propio fichero NED o a través del fichero de configuración *omnetpp.ini*. La ventaja de hacerlo a través del fichero de configuración es que esta opción permite definir distintas configuraciones de red de modo que en cada una de ellas se asignan las rutas de ficheros de redes distintas. De este modo, es posible generar nuevos escenarios sin que sea necesario compilar todo el proyecto una y otra vez. En la figura 4.5 se muestra un ejemplo del

fichero de configuración *omnetpp.ini* en el que se incluyen las configuraciones *Network1* y *Network2*. Así, en el momento de lanzar la simulación, en función de la configuración escogida se creará un escenario distinto.

```
[Config Network1]
**.nodesFile = "networks/nodes.txt"
**.linesFile = "networks/LVlines.txt"
**.busbarsFile = "networks/LVbusbars.txt"
**.connectionsFile = "networks/connections.txt"
**.lineConnections = "networks/LineConnections.txt"
**.busbarConnections = "networks/BusbarConnections.txt"

[Config Network2]
**.nodesFile = "networks/nodes2.txt"
**.linesFile = "networks/LVlines2.txt"
**.busbarsFile = "networks/LVbusbars2.txt"
**.connectionsFile = "networks/connections2.txt"
**.lineConnections = "networks/LineConnections2.txt"
**.busbarConnections = "networks/BusbarConnections2.txt"
```

Figura 4.5: Asignación de ficheros de texto que contienen información de la red desde *omnetpp.ini*

Para poder construir la red a partir de estos ficheros de texto es necesario implementar una clase que se encarga de leer la información de los ficheros, de crear las instancias de los módulos y de establecer las conexiones entre ellos. En el modelo desarrollado, la clase encargada de realizar estas tareas se conoce como "ConstructorDeRed". Por otro lado, tal y como se observa en el ejemplo mostrado en la figura 4.5, para la creación de los escenarios se han empleado los siguientes ficheros de texto:

- ***nodesFile***. Se trata del fichero en el que se incluyen los tipos de nodos que conforman la red (*nodetype*), así como otros parámetros asociados a cada uno de ellos como puede ser su nombre (*name*), identificador (*id*), la coordenadas (x, y), a qué línea de alimentación o *LV feeder* está conectado (*lineID*) y fabricante del nodo (*manufacturer*). La figura 4.6 muestra un ejemplo de este tipo de fichero.

```
#id name  nodetype                   lineID  x   y  manufacturer
0   BN    primenet.node.Concentrator  0      50  200   ziv
1   SN1   primenet.node.PLCMeter      1      200 220   ziv
2   SN2   primenet.node.PLCMeter      1      300 270   ziv
... ...         ...                  ...    ... ...   ...
19  SN19  primenet.node.PLCMeter      2      450 220   ziv
20  SN20  primenet.node.PLCMeter      2      550 270   ti
21  SN21  primenet.node.PLCMeter      2      450 220   ziv
... ...         ...                  ...    ... ...   ...
```

Figura 4.6: Fichero que contiene el listado de nodos que componen la red (*nodesFile.txt*)

- ***linesFile***. En este fichero se incluye la información de las líneas de alimentación de las que se compone la red. En él se incluye el identificador único de cada línea (*id*), el identificador del bus al cual se conecta cada línea (habrá un bus por cada transformador) (*busID*), a qué distancia del nodo base se conecta la línea al bus (*distSS*), el tipo de módulo a partir del cual se van a crear las líneas (*type*) y las posiciones de los puntos de suministro de cada línea (*positions*). La figura 4.7 muestra un ejemplo de un fichero de tipo *linesFile*.

```
#id busID distSS type                          positions
1   0     250    primenet.linklayer.LVLine   0_20_70_120_ ...
2   0     500    primenet.linklayer.LVLine   0_20_70_120_170_...
3   0     750    primenet.linklayer.LVLine   0_20_70_120_...
4   0     1000   primenet.linklayer.LVLine   0_20_70_120_170_...
... ...   ...             ...                    ...
```

Figura 4.7: Fichero que contiene el listado de líneas de alimentación que componen la red (*linesFile.txt*)

- ***busbarsFile***. En este fichero se incluye la información de los buses que hay en la red. En cada red habrá al menos un bus conectado al nodo base. Si además, hay uno o más nodos auxiliares en la red (uno por cada transformador extra), habrá un bus conectado a cada nodo auxiliar. En el fichero de texto en el que se incluyen estos datos, por un lado aparece el identificador único de cada bus (*id*), el módulo a partir del cual se crean las instancias de bus (*type*), la longitud del bus (*length*) (parámetro que se emplea para dibujar los buses) y las posiciones del bus donde irán conectadas las líneas de alimentación (*positions*). La figura 4.8 muestra un ejemplo de un fichero de tipo *busbarsFile*.

```
#id    type                          length  positions
0      primenet.linklayer.LVBusbar 1250   0_250_500_750_1000
...              ...                  ...          ...
```

Figura 4.8: Fichero que contiene el listado de buses de alimentación que componen la red (*busbarsFile*)

- ***connectionsFile***. En este fichero se indica a qué línea va conectado cada nodo de la red (*lineID*) y en que posición irá conectado (*position*) tal y como se muestra en el ejemplo de la figura 4.9.

```
# nodeID    lineID    position
1           1         1
2           1         2
3           1         3
...         ...       ...
19          2         1
20          2         2
21          2         3
...         ...       ...
```

Figura 4.9: Fichero que define las conexiones entre nodos y líneas (*connectionsFile.txt*)

- ***lineConnections***. En este fichero se indica a qué bus (*busID*) y en qué posición del mismo (*position*) se conecta cada línea (ver figura 4.10).

```
#lineID   busID   position
1         0       1
2         0       2
3         0       3
4         0       4
```

Figura 4.10: Fichero que define las conexiones entre líneas y buses (*lineConnections.txt*)

- **busbarConnections**. Este fichero incluye la lista de los nodos que están conectados directamente a un bus (como por ejemplo el nodo base o nodos auxiliares) y cuenta con los siguientes campos: identificador de nodo (*nodeID*), el bus al que se conecta (*busID*) y la posición en la que lo hace (*position*) (ver figura 4.11).

```
#nodeID   busID position
0         0     0
```

Figura 4.11: Fichero que define las conexiones de los nodos conectados a un bus directamente (*busbarConnections.txt*)

Tal y como se explicará mas adelante en este capítulo, la ubicación física de los elementos (nodos, líneas, etc.) que componen la red no tienen por qué representar la ubicación real de los mismos, ya que para determinar los destinos de las tramas transmitidas el modelo de simulación desarrollado se basa en el fichero con la topología lógica más común de la red a simular. Sin embargo, la información de la ubicación física de los elementos permite realizar una representación gráfica de la red y a su vez, permite introducir un retardo en la recepción de tramas por los distintos nodos que componen dicha red. Así, la

figura 4.12 muestra la apariencia de un ejemplo de una subred PRIME creada a partir de los ficheros descritos.

Figura 4.12: Apariencia de un ejemplo de subred PRIME en el modelo de simulación

4.2.3 Fichero con la topología lógica más común

La mayor parte de modelos de simulación analizados en el capítulo 3 del estado del arte se centran en la caracterización física del canal PLC a partir del cual se determinan los destinos de cada trama transmitida. Mediante dicha caracterización para poder reproducir el comportamiento de una red real conocida sería necesario conocer las distancias entre cada par de nodos (emisor-receptor). Sin embargo, en la mayoría de los casos, las operadoras eléctricas no disponen de esta información o no es 100 % fiable [SBA+13].

El modelo de simulación desarrollado en este trabajo de investigación emplea la información de la topología lógica más común para reproducir el comportamiento de una red real. Esta información es obtenida a partir de un proceso de monitorización de la red real a simular realizado mediante el nodo base de dicha red. El objetivo de dicho proceso de monitorización, es el de entender el comportamiento de la red [FOSEUG+13, SBA+13]. La figura 4.13 muestra un ejemplo de una topología lógica de una subred PRIME proporcionada por el nodo base.

La topología lógica más común de la red se almacena en un fichero XML donde se enumeran los padres más comunes y el nivel en la jerarquía más común de cada nodo. De este modo, un nodo que dependa directamente del nodo base se registrará a través de este, mientras que un nodo que dependa de un nodo de servicio, tendrá que esperar a que dicho nodo de servicio se haya registrado y convertido en *Switch* para poder registrarse a través de él.

A la hora de generar el fichero XML que contiene la topología lógica más común, la dirección física o MAC de cada nodo de servicio de la subred empleado en el proceso de registro (ver apartado 2.2.3.2 del capítulo 2) ha sido reemplazado por un identificador de nodo numérico tal y como muestra la figura 4.14. El identificador de cada nodo se obtiene del campo *id* del fichero *nodesFile.txt* (el 0 está reservado para el nodo base) y será el mismo que asignará el nodo base a cada nodo de servicio en los procesos de registro y promoción. De este modo, el direccionamiento explicado en el estándar se simplifica de manera considerable.

Por otro lado, en cuanto a los niveles en la jerarquía, es importante mencionar que los nodos que están conectados directamente al nodo base tienen el nivel 0, mientras que el nivel de cualquier nodo que no esté conectado al nodo base tiene el nivel de su *Switch* inmediato más uno (ver 2.2.3.2.1 del capítulo 2).

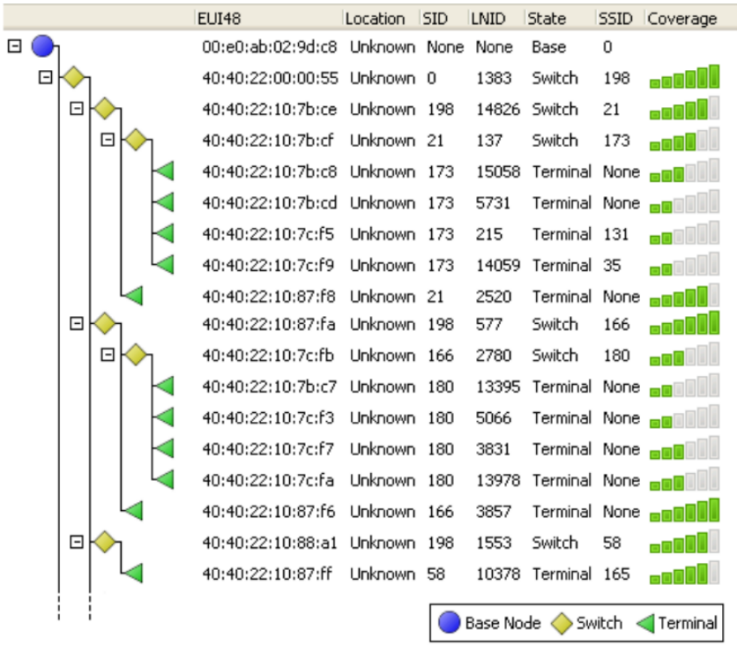


Figura 4.13: Ejemplo de una topología lógica de una subred PRIME proporcionada por el nodo base.

```
<?xml version="1.0"?>
<root>
  <node id="1">
    <state>
      <parent>0</parent>
      <level>0</level>
    </state>
  </node>
  ...
  <node id="22">
    <state>
      <parent>1</parent>
      <level>1</level>
    </state>
  </node>
  ...
</root>
```

Figura 4.14: Ejemplo de un fichero XML que contiene la topología lógica más común de una subred

4.2.4 Líneas y buses

Las líneas y buses de alimentación son unos componentes de gran importancia dentro del modelo de simulación ya que son los encargados de transportar los paquetes que transmiten los distintos nodos de la subred.

En las subredes PRIME reales, no todos los nodos pueden comunicarse directamente con todos los demás nodos. Una de las razones de esto puede ser la elevada distancia entre los nodos, o además, también puede deberse al hecho de que los nodos estén conectados a distintas fases de la línea. Como ya se ha adelantado, las operadoras eléctricas no siempre conocen la ubicación de cada uno de los nodos que hay en la red. Por esta razón, para determinar si un nodo puede comunicarse directamente con otro nodo, se ha implementado un algoritmo basado en los niveles lógicos de los nodos.

Para ello, en primer lugar, en el fichero de configuración *omnetpp.ini*, se ha definido un parámetro llamado *colllisionDomain* que determina el número máximo de niveles lógicos que podrá atravesar un paquete cuando es transmitido. Normalmente, los nodos podrán comunicarse con nodos de su mismo nivel o con aquellos nodos con un nivel inmediatamente superior o inferior en la jerarquía. En ocasiones, también ocurre que nodos que cuentan con una diferencia de hasta 2 niveles en la jerarquía puedan comunicarse. Para ello, en el parámetro *collisionDomain* se fija un valor que hace referencia a la diferencia de niveles máxima que puede haber entre cada emisor y receptor. Este valor puede ser cualquier valor mayor que 1, pero según la experiencia de Iberdrola en pruebas de campo este valor deberá ser como mucho un 2.

Así, cuando un nodo transmite un paquete a la línea, cuando el paquete llega a cada punto de suministro de la misma, se comprobará el nivel lógico del nodo conectado en ese punto y se calculará un número aleatorio siguiendo la

ecuación 4.1, lo que servirá para determinar si el paquete debe ser recibido por el nodo o no. Después, se calculará la diferencia de niveles lógicos entre el nodo transmisor y el posible receptor, y si este valor es menor o igual al valor aleatorio calculado, la línea le transmitirá una copia del paquete a dicho receptor. En caso contrario, el nodo no recibirá nada y el paquete seguirá su camino a través de la línea.

$$levelDifference = rand()\%collisionDomain + 1 \tag{4.1}$$

Es importante mencionar que para que la línea pueda saber en qué nivel lógico se encuentra cada nodo, le realizará una consulta al módulo *Network Manager* ya que este módulo conoce el estado global de la red en todo momento. El funcionamiento del *Network Manager* se explicará más adelante en la sección 4.2.6 de este capítulo.

Para facilitar la comprensión del algoritmo descrito, la figura 4.15 muestra un ejemplo teórico. En dicho ejemplo, el nodo transmisor del paquete (contiene una flecha indicativa) se encuentra en el nivel 1 de la jerarquía. Cuando el nodo envía el paquete, éste se propagará en ambas direcciones de la línea. Cuando el paquete llegue a cada uno de los puntos de suministro (con el correspondiente retardo), la línea consultará al *Network Manager* el nivel del nodo que está conectado en ese punto y calculará un número aleatorio comprendido entre el 1 y el valor del parámetro *collisionDomain* (en caso de que su valor sea distinto de 1). De este modo, si la diferencia de niveles del nodo transmisor del paquete y el nodo conectado en dicho punto es menor o igual al número aleatorio calculado, el paquete será recibido por el nodo. En cambio, si la diferencia es mayor que el número calculado, el paquete no será transmitido por dicho punto de suministro. Por lo tanto, tal y como se aprecia en la figura, los nodos que recibirán el paquete son el nodo base y los nodos de servicio 1, 2 y 4, ya que su diferencia de niveles con respecto al nodo transmisor es menor o igual a 1. Sin embargo, como se puede observar, el nodo 5, que se encuentra en el nivel 3 de la jerarquía, no recibe el paquete ya que en este ejemplo, el número aleatorio calculado por la línea para este punto de suministro es 1 y la diferencia de niveles entre los nodos es de 2.

En el segundo ejemplo mostrado en la figura 4.16, el nodo transmisor es el nodo 2 el cual se encuentra en el nivel 1 de la jerarquía. En este caso, todos los nodos de la línea reciben el paquete, incluido el nodo 5 de nivel 3 ya que el número aleatorio calculado por la línea para este punto ha sido un 2. Por lo tanto, aunque la diferencia de niveles entre el nodo 2 y el 5 sea de 2, el paquete ha podido llegar al nodo 5.

Por otro lado, en el tercer ejemplo (figura 4.17) el nodo transmisor es el nodo 5 de nivel 3. El paquete llega al nodo de servicio 4 que corresponde al nivel 2, y por lo tanto la diferencia de niveles de estos dos nodos es de 1. Por otro lado, el paquete también llega al nodo 3 de nivel 1, puesto que la línea ha calculado

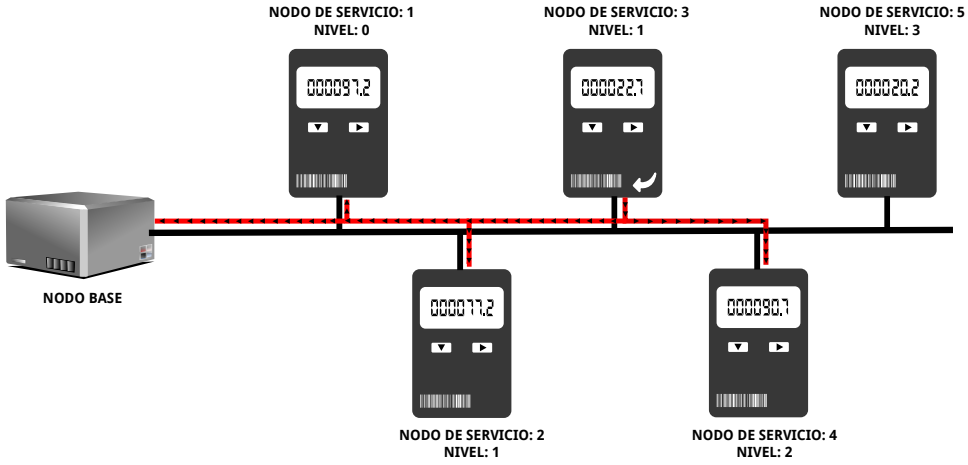


Figura 4.15: Ejemplo 1 de transmisión de paquetes por la línea

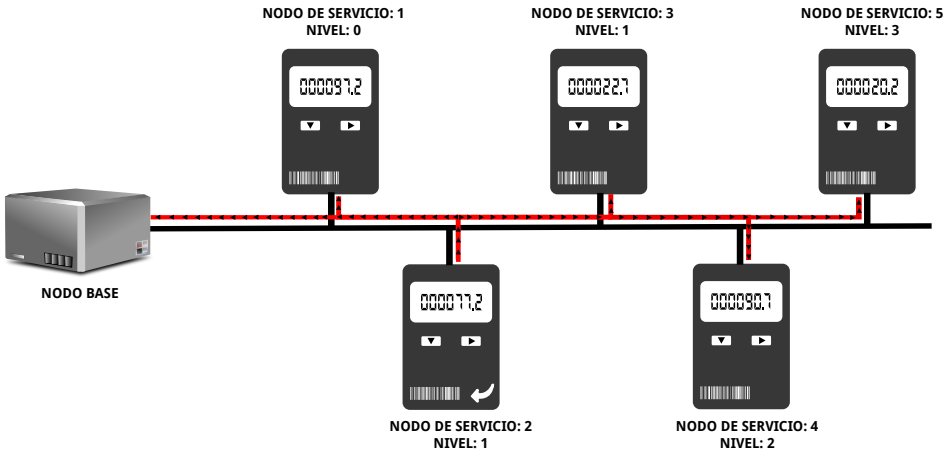


Figura 4.16: Ejemplo 2 de transmisión de paquetes por la línea

que la diferencia de niveles entre los nodos ha de ser menor o igual a 2. Sin embargo, el paquete no llega al nodo de servicio 2 a pesar de que al igual que el nodo 3 también es de nivel 1, ya que para este caso la línea ha determinado que la diferencia de niveles entre los nodos ha de ser menor o igual a 1.

Por último, es importante mencionar que el modelo de simulación también tiene en cuenta los retardos de propagación entre los nodos. Dichos retardos se calculan en función de las características físicas de los cables que nos darán la velocidad de propagación, la distancia que hay entre los nodos y el tamaño de los paquetes. Como ya se ha adelantado, los valores de distancia empleados en este modelo de simulación no tienen por qué ser los reales, ya que su principal

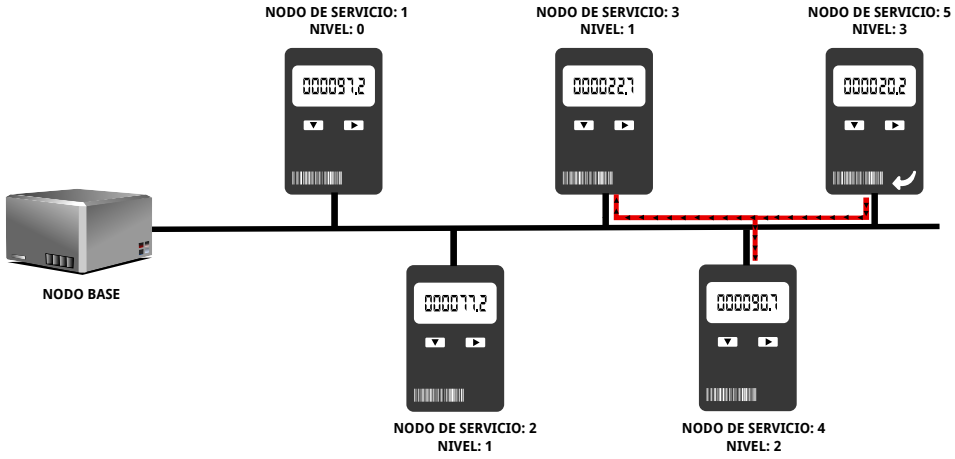


Figura 4.17: Ejemplo 3 de transmisión de paquetes por la línea

función es, por un lado, poder llevar a cabo una representación gráfica de la red a simular, y por otro lado, introducir un retardo en la recepción de las tramas de modo que éstas no sean recibidas al mismo tiempo.

4.2.5 Nodos

En la presente sección se analizará la estructura de los nodos que componen la red. Siguiendo el esquema del *stack* PRIME tanto los nodos de servicio como el nodo base contarán con la estructura de capas que se muestra en la figura 4.18. Como se aprecia en dicha figura, un nodo PRIME es un módulo compuesto por los módulos simples *APP*, *CL* y *PHY* y el módulo compuesto *MAC*. El módulo *MAC* por su parte esta compuesto por los módulos simples *Control*, *Switching* y *CSMA/CA*. La figura 4.19a muestra la estructura que tiene un nodo de este tipo en el entorno gráfico de OMNeT++, mientras que la figura 4.19b muestra la estructura del módulo compuesto MAC que forma parte de dicho nodo. A continuación, se incluye una breve explicación de las funciones que desempeña cada parte.

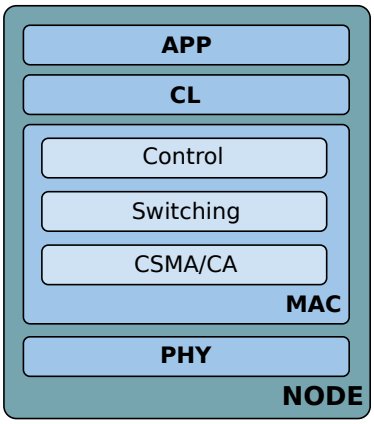
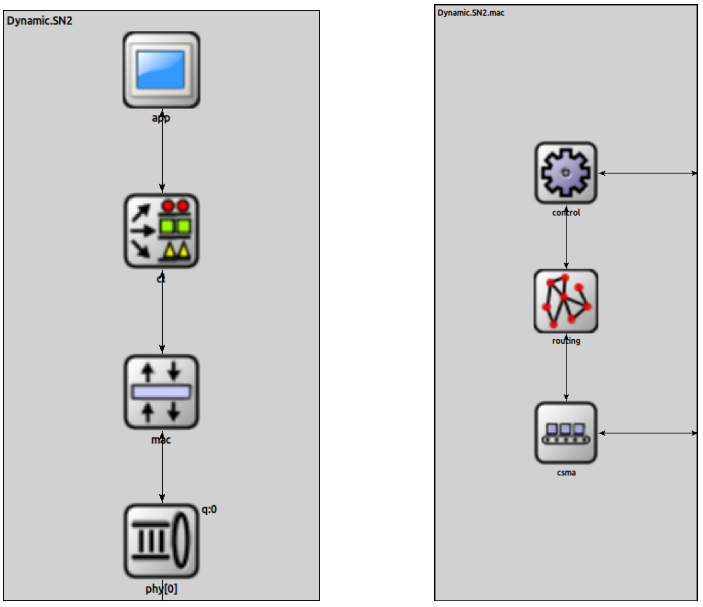


Figura 4.18: Estructura de capas de los nodos PRIME



(a) Estructura de nodo (b) Estructura MAC

Figura 4.19: Estructura general y de capa MAC de los nodos PRIME

- **APP**. Capa de aplicación. En este trabajo de investigación se ha implementado la aplicación de actualización *firmware*.

- **CL**. Capa de convergencia. Esta capa clasifica el tráfico asociándolo con su conexión MAC correcta. En este caso, se ha implementado la subcapa de convergencia *NULL* que proporciona una ruta directa desde la capa MAC a las superiores.

- **MAC**. Capa de control de acceso al medio. La capa MAC proporciona funcionalidades básicas MAC de acceso al sistema como la asignación de ancho de banda, la creación/mantenimiento de las conexiones y la resolución de la topología.

  - *Control*. Este submódulo se encarga de procesar los paquetes de control que se intercambian en las típicas operaciones de capa MAC como son la gestión del registro, gestión de la promoción, establecimiento de la conexión, indicación de *slot* de baliza, mecanismo *KeepAlive*, etc.
  - *Switching*. Este submódulo es el encargado de realizar las operaciones de *switching* o conmutación. Se encarga de decidir si los paquetes recibidos por el nodo se dirigen al propio nodo de modo que se retransmiten a las capas superiores, a un nodo que se ha registrado a través de este nodo y deben ser retransmitidos a la línea, o deben ser descartados.
  - *CSMA/CA*. Este submódulo implementa el esquema de acceso al canal CSMA/CA explicado en la sección 2.2.3.4 del capítulo 2.

- **PHY**. Capa física. La capa PHY de PRIME trabaja en la banda CENELEC-A [CEN93] en el rango de frecuencias de 41 kHz a 89 kHz. Utiliza la técnica de multiplexación OFDM con diferentes esquemas de modulación: DBPSK, DQPSK o D8PSK con la opción de emplear o no el mecanismo FEC. Además, este submódulo comprueba si el paquete recibido ha sufrido una colisión y realiza la pérdida aleatoria de paquetes para reproducir el efecto de la degradación SNR en la línea [PRI12].

4.2.5.1 Capa PHY

Como ya se ha adelantado en apartados anteriores, la capa PHY de los nodos que componen una subred PRIME se encarga de distintas funciones. Por un lado, cuando un nodo recibe un paquete, la capa física del modelo se encarga de verificar la integridad de dicho paquete. Para ello, en el modelo de simulación desarrollado, el nodo obtendrá el identificador único de cada paquete y consultará al módulo *NetworkManager* si el paquete con dicho identificador ha sufrido una colisión. En el apartado 4.2.6.2 se explica el mecanismo de reproducción de colisiones implementado en detalle.

Por otro lado, para reproducir el efecto de la degradación SNR de la línea se ha implementado un mecanismo de descarte aleatorio de paquetes. Para ello, en el fichero de configuración *omnetpp.ini* se ha definido el parámetro *packetLoss* que fija el porcentaje de pérdida de paquetes de cada nodo. De este modo, cuando un nodo reciba un paquete, el submódulo PHY del nodo calculará un número

aleatorio comprendido entre 0 y 100. Así, si el valor calculado tiene un valor menor o igual al fijado por el parámetro *packetLoss* el paquete será descartado. En caso contrario, el paquete será transmitido a las capas superiores del nodo.

4.2.5.2 Capa MAC

El módulo *MAC* que forma parte de los nodos, es un módulo compuesto que se compone de tres módulos simples que se detallan a continuación (figura 4.19b).

- **CSMA/CA**.

  Este módulo es el encargado de ejecutar el algoritmo CSMA/CA el cual se ha implementado tal y como se ha explicado en el apartado 2.2.3.4 del capítulo 2. Sin embargo, existen dos pasos del algoritmo requieren de especial atención.

  Por un lado, cuando el nodo ejecuta el algoritmo CSMA/CA, en primer lugar, se calculan el periodo de *backoff*, el número de iteraciones de comprobación del canal a realizar (en función de la prioridad del paquete) y el tiempo necesario para la transmisión del paquete, de modo que el nodo pueda saber si en el tiempo que queda de SCP es posible enviar el paquete que desea transmitir.

  El tiempo necesario para la transmisión del paquete dependerá del tamaño del mismo y del esquema de modulación utilizado, tal y como se explicará más adelante en el apartado 4.2.7 de este capítulo.

  La longitud del SCP así como lo que queda de dicho periodo, se calcula a partir de la información de la última baliza recibida. Una baliza o trama BPDU tiene los campos *BCN.CNT*, *BCN.SLT* y *BCN.CFP* (ver A.2.3 del apéndice A) mediante los cuales un nodo de servicio puede saber cuándo comienza y termina el periodo *SCP* así como lo que queda de él cuando se dispone a transmitir un paquete.

  Para explicar cómo se realiza este cálculo en el modelo de simulación, a continuación se incluye un ejemplo en el que un nodo de servicio recibe una baliza de su nodo *Switch* padre con los siguientes valores: $BCN.CNT = 5$, $BCN.SLT = 2$ y $BCN.CFP = 0$. Para calcular cuándo comienza el periodo SCP, se sabe que la baliza recibida tiene asignado el *slot* número 2 de la trama ($BCN.SLT = 2$) por lo que teniendo en cuenta que el *slot* 0 está reservado para el nodo base, la baliza recibida ocupa el tercer *slot* de la trama. Por lo tanto, debido a que en la trama hay 5 *slots* ocupados ($BCN.CNT = 5$) y cada baliza tiene una longitud de 4 símbolos, transcurrirán 8 símbolos (2 *slots*) desde la recepción de la última baliza y antes del comienzo del SCP.

Por otro lado, para conocer el tiempo restante del periodo SCP, el nodo tendrá que fijarse en el valor del campo *BCN.CFP*. En este ejemplo, el valor de dicho campo es cero por lo que no habrá periodo CFP y el SCP durará hasta que finalice la trama. Por lo tanto, sabiendo que la duración total de la trama es de *MACFrameLength* (276 símbolos), que cada baliza tiene una duración de *MACBeaconLength* (4 símbolos) (ver A.2.4 del apéndice A) y esta trama tiene ocupados 5 *beacon-slots*, la duración del SCP en símbolos será de 256 tal y como muestra la ecuación 4.2.

$$276 \text{ símbolos} - 4 \frac{\text{símbolos}}{\text{baliza}} \times 5 \text{ balizas} = 256 \text{ símbolos} \tag{4.2}$$

De este modo, cuando el módulo CSMA/CA del nodo recibe un paquete desde las capas superiores, en primer lugar, comprueba en qué periodo de la trama MAC se ha recibido dicho paquete. Si el periodo SCP no ha comenzado calculará y esperará el tiempo restante hasta su comienzo, y cuando éste comience se pondrá en marcha el mecanismo CSMA/CA. En este caso, la duración restante del SCP será de 256 símbolos. En cambio, si el paquete llega durante el SCP, el tiempo restante del SCP se calculará a partir de tiempo que ha transcurrido desde el comienzo de dicho periodo y lo que queda hasta que termine la trama.

Por otro lado, para poder implementar el algoritmo CSMA/CA, los nodos deben ser capaces de realizar comprobaciones de canal para saber si está libre para transmitir. Para poder reproducir este comportamiento de los nodos, teniendo en cuenta que en el modelo de simulación no existe un medio físico real, el nodo realizará una consulta al módulo *Network Manager* y éste comprobará el estado del canal en ese instante. El mecanismo empleado por el *Network Manager* para determinar si el canal está libre para transmitir se explicará más adelante en el apartado 4.2.6.1 de este capítulo.

- **Switching**.

  El módulo *Switching* será el encargado de decidir si los paquetes recibidos por el nodo han de ser descartados, transferidos a capas superiores del módulo y/o retransmitidos a la línea. En la sección 2.2.3.3 del capítulo 2 se explican los distintos mecanismos de *switching* que se han implementado en este módulo: *switching* de paquetes *unicast*, *broadcast* y *multicast*.

- **Control**.

  Como ya se ha adelantado, este módulo es el encargado de procesar los paquetes de control necesarios para llevar a cabo las típicas operaciones de capa MAC como el registro, promoción, establecimiento de conexión,

asignación de *slot* de balizas, mecanismo *Keep-Alive*, etc. que se explicarán a continuación.

- Mecanismo de registro. El mecanismo de registro de un nodo de servicio ya ha sido explicado en el apartado 2.2.3.6.1 del capítulo 2 y a continuación, se incluyen los detalles de la implementación de este mecanismo en el modelo de simulación.

Al comienzo de la simulación, se carga el fichero XML que contiene la topología lógica más común de la red y cada nodo de servicio extrae la parte del fichero XML que le corresponde. En el fichero aparecen el padre o padres más comunes del nodo servicio así como el nivel más común del nodo en la jerarquía.

Al principio, todos los nodos de servicio se encuentran en el estado *Desconectado* y tratarán de registrarse. Un nodo de servicio que se encuentra en este estado se queda a la espera de escuchar balizas o tramas BPDUs enviadas por alguno de los nodos padre indicados por el fichero XML, de manera que pueda iniciar el proceso de registro a través de alguno de ellos. El nodo de servicio esperará durante *macMinSwitchSearchTime* segundos (ver A.3.1 del apéndice A) para ver si escucha alguna de las balizas esperadas (se activará un temporizador de esta duración). Pasado este tiempo si el nodo no ha recibido ninguna baliza comenzará a enviar paquetes PNPDU para solicitar que algún nodo vecino en el estado *Terminal* pida ser promocionado y se convierta en *Switch*. En cambio, si durante el tiempo de espera el nodo recibe un BPDU por parte de alguno de sus padres, se cancelará el temporizador de espera de balizas y el nodo enviará el paquete de petición de registro correspondiente (*REG_REQ*) al nodo base tal y como indica el estándar. Cuando el nodo de servicio envía un paquete de petición de registro, inicia un temporizador de espera a la respuesta por parte del nodo base. La duración de este temporizador se configura desde el fichero de configuración *omnetpp.ini* a través del parámetro *macCtlReTxTimer* (ver A.3.1 del apéndice A). Si durante el tiempo marcado por dicho parámetro, el nodo no recibe respuesta por parte del nodo base, el nodo de servicio vuelve a enviar el paquete de petición de registro. En cambio, si el nodo de servicio recibe la respuesta de aceptación de registro (*REG_RSP*) por parte del nodo base, el nodo de servicio pasará al estado *Terminal* y cancelará dicho temporizador. El nodo de servicio, por su parte, le enviará el paquete de confirmación *REG_ACK* al nodo base para comunicarle que el registro del nodo ha sido completado. Es importante mencionar que el cambio de estado del nodo de servicio se comunica también al *Network Manager*, que es el encargado de conocer el estado de todos los elementos de la subred.

- Mecanismo de promoción. Cuando un nodo de servicio en el estado *Terminal* recibe tramas de tipo *PNPDU* por parte de un nodo que se encuentra en estado *Desconectado* y no consigue comunicarse con ningún *Switch*, el nodo *Terminal* podrá enviar paquetes de solicitud de promoción (*PRO_REQ*) al nodo base de modo que pueda convertirse en *Switch*.

El algoritmo mediante el cual un nodo de servicio en el estado *Terminal* toma la decisión de atender o no a la petición del nodo de servicio en estado *Desconectado* no se concreta en el estándar (aunque si se dan algunas pautas) y cada fabricante de equipos lo implementa a su manera. Por ejemplo, el estándar establece que como máximo un nodo de servicio en el estado *Terminal* podrá ignorar hasta un máximo de *MACMaxPRNIgnore* (ver A.3.1 del apéndice A) paquetes *PNPDU* de un mismo dispositivo. La recepción de múltiples *PNPDU*s de un mismo dispositivo indica que no hay ningún otro dispositivo en las inmediaciones del nodo en estado *Desconectado* que pueda ayudarle a unirse a la subred mediante su promoción.

El algoritmo de decisión implementado en el modelo de simulación funciona de la siguiente manera. En primer lugar, se fija el porcentaje de paquetes *PNPDU* a ignorar por los nodos en estado *Terminal* mediante el parámetro *ignorePNPDUpercentage* declarado en el fichero de configuración (*omnetpp.ini*). Así, si por ejemplo el porcentaje de aceptación de paquetes es de un 25 %, cada vez que llega un paquete *PNPDU* a un nodo en el estado *Terminal* éste calculará un número aleatorio entre 0 y 100. Si el número calculado es menor que 25 el paquete será procesado y por tanto el nodo de servicio enviará un paquete de petición de promoción al nodo base. En cambio, si el número calculado se encuentra entre el 25 y 100, el paquete será descartado como máximo hasta *MACMaxPRNIgnore* número veces. Por lo tanto, en caso de que los paquetes *PNPDU* provenientes de un mismo nodo de servicio hayan sido descartados *MACMaxPRNIgnore* número de veces, el paquete será procesado y el nodo en estado *Terminal* finalmente enviará la petición de promoción al nodo base.

Como se puede ver, es posible que más de un nodo de servicio en estado *Terminal* solicite ser promocionado para poder ayudar al nodo en estado *Desconectado*. Ante esta situación, cuando el nodo base recibe la primera solicitud de promoción iniciará un temporizador que fijará el tiempo durante el cual se almacenarán las solicitudes que vayan llegando, antes de comenzar a procesarlas y escoger el nodo a promocionar. El modo en el que el nodo base toma la decisión de qué nodo va a promocionar no está especificada por el estándar, por lo que, cada fabricante implementa su propio algoritmo. En los dispositivos reales de una subred real, el paquete de petición de promoción *PRO_REQ* incluye un parámetro llamado *LQI* que indica la calidad del paquete *PNPDU* enviado por el nodo en

estado *Desconectado*. Algunos de los algoritmos implementados en nodos base reales toman la decisión en base a este parámetro. En el modelo de simulación en cambio, este parámetro ha sido sustituido por el identificador del nodo en estado *Desconectado*. De este modo, cuando las distintas solicitudes de promoción lleguen al nodo base, éste las procesará y mediante la consulta al fichero XML de topología lógica más común, escogerá el nodo de servicio a promocionar. El nodo escogido recibirá el paquete de respuesta *PRO_REQ_B*, mientras que el resto de nodos no recibirán nada para no saturar la red. Por lo tanto, el nodo o los nodos que reciban este paquete cambiarán su estado de *Terminal* a *Switch*, y enviarán el paquete de confirmación *PRO_ACK* al nodo base a modo de confirmación (ver apartado 2.2.3.6.3 del capítulo 2). La figura 4.20 muestra el diagrama de funcionamiento del algoritmo de promoción implementado.

- Asignación de balizas. Una vez que el nodo de servicio haya cambiado su estado del estado *Terminal* a *Switch* y el nodo base reciba el paquete de confirmación correspondiente (*PRO_ACK*) por parte del nodo de servicio, el nodo base deberá indicar al nuevo *Switch* el *slot* en el que deberá transmitir su trama BPDU o baliza.

Para ello, en primer lugar, el nodo base deberá reservar un *slot* de la trama dedicado a la emisión de balizas para dicho *Switch*. El algoritmo o mecanismo de asignación de balizas no se detalla en la especificación, aunque existen algunas condiciones a cumplir, por lo que cada fabricante de nodos base implementará su propio algoritmo. En el modelo de simulación desarrollado la manera en la que se asignan dichos *slots* se detalla a continuación.

Si el número total de *Switches* en la red (contando al recién promocionado) es menor que 32, la siguiente baliza a asignar ocupará el *slot* 2, mientras que si es mayor que 32 y menor o igual a 64 ocupará el *slot* 4. De este modo, cuando la red cuenta con más de 32 *Switches* al dejar un *slot* libre entre el envío de balizas, nos aseguramos de que no ocurran colisiones entre balizas.

Por otro lado, el algoritmo desarrollado prevé que cada *Switch* exceptuando al nodo base, emita una baliza cada 32 tramas o lo que es lo mismo, 1 baliza por cada supertrama (ver definición de supertrama en el apartado 2.2.3.4.3 del capítulo 2). Por lo tanto, el cálculo realizado para escoger el número de secuencia de la trama en la que se deberá transmitir la baliza se muestra a en la figura 4.21.

Después, el nodo base enviará un paquete de tipo *BSI_IND* al nuevo *Switch* para que conozca cuándo tiene que transmitir su baliza. El *Switch* por su parte contestará con un paquete de tipo *BSI_ACK* a modo de confirmación.

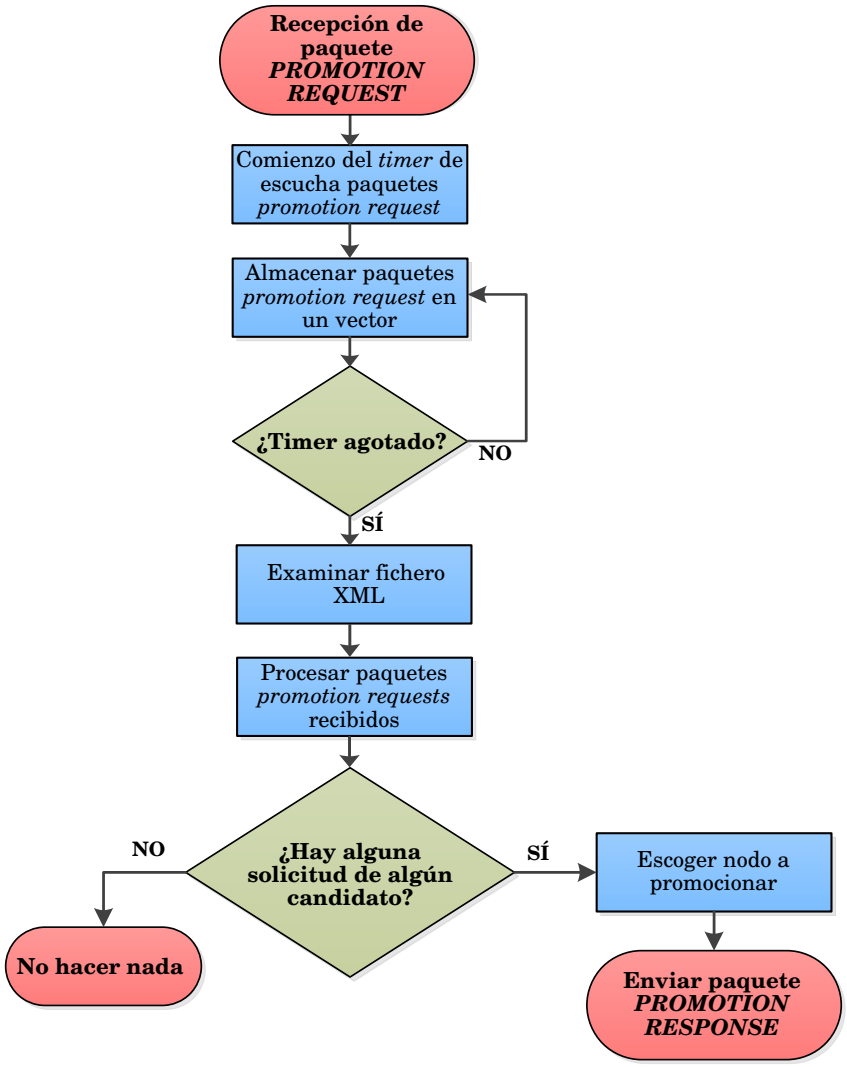


Figura 4.20: Diagrama de flujo que representa el algoritmo de promoción

- Envío de balizas. Cuando un nodo que estaba en el estado funcional *Terminal* pasa al estado de *Switch*, como ya se ha adelantado, el nodo base le asigna el *slot* en el que deberá transmitir su baliza. Una vez asignado este *slot*, el nodo base le enviará un paquete de tipo *BSI_IND* al nuevo *Switch* para que éste sepa cuándo tiene que transmitir su baliza. Cuando al *Switch* le llegue el turno de transmitir su baliza, para ahorrar un elevado número de eventos y agilizar así las simulaciones, en lugar de transmitir dicho paquete a través de los módulos de líneas y buses, se ha optado por simplificar este mecanismo sustituyéndolo por el envío de señales. De este modo, cuando al nodo base o al nodo *Switch* le toque transmitir dicha

```
numeroDeSecuencia = (16 x resto(numeroDeSwitches/2) + numeroDeSwitches
     - redondeoHaciaArriba((numeroDeSwitches + 1)/2)) - (8 x (slot-2))
```

Figura 4.21: Algoritmo empleado para asignar el número de secuencia para la transmisión de balizas

baliza, llamará a la función *emit(beacon,frame)* (*beacon* es el nombre del tipo de señal y *frame* es un puntero a objeto que contiene la información de la trama *BPDU*) para emitir la señal que hace las veces de baliza. A su vez, el nodo *Switch* también le notificará al *Network Manager* del envío de dicha baliza de modo que quede registrado. Así, cada vez que algún nodo emita una baliza, se emitirá una señal global a nivel de simulación, y los nodos que dependen de dicho nodo, indicado por el fichero XML, la procesarán y actuarán en consecuencia. En la sección 4.2.8 de este capítulo se explicará con más detalle el funcionamiento del mecanismo de señales que ofrece OMNeT++.

En las simulaciones por eventos discretos, el tiempo que duran las simulaciones depende del número de eventos que ocurren en dicha simulación así como de las prestaciones de la máquina en la que se lleva a cabo. Cuando se envía un paquete a la línea, se programará un evento de simulación por cada punto de suministro para que el paquete llegue a cada uno de los destinos con el retardo correspondiente. Este retardo se calculará en función de la distancia entre nodos, la velocidad de propagación del medio y el tamaño del paquete.

Como ya se ha explicado en el apartado del capítulo 2, el nodo base envía una baliza en cada trama, es decir cada 276 símbolos (618 ms aproximadamente). Si las balizas enviadas por el nodo base llegasen a 20 nodos de servicio, cada vez que el nodo base envía una baliza se sumarían 20 eventos a la cola de eventos de simulación. Si una simulación tiene una duración de 3 horas (tiempo de simulación), durante ese tiempo el nodo base enviaría 17.475 balizas, por lo que si por cada baliza se acumulan 20 eventos, al final de la simulación se habrán procesado 349.500 eventos sólo por el envío de dicha baliza. Si además, existen más *Switches* en la red, y cada una de las balizas se transmite cada 32 tramas (cada 19 segundos aproximadamente) acumularíamos también *x* número de eventos cada vez que un *Swicth* transmita cada baliza.

Otro de los motivos por los que se ha escogido el mecanismo de señales para simular el envío de balizas es que teniendo en cuenta que las balizas tienen sus *slots* reservados y no utilizan el periodo SCP la probabilidad de que colisionen con otros paquetes es menor. Sin embargo, el algoritmo de reproducción de colisiones que se explicará en el apartado 4.2.6 también

toma en cuenta las colisiones entre balizas, y entre paquetes y balizas, como por ejemplo puede ocurrir con paquetes de tipo *PNPDU* que son enviados por nodos de servicio que no conocen la estructura de la trama.

- Seguimiento de balizas. Cuando un nodo de servicio en el estado *Desconectado* recibe una baliza, en caso de que reciba varias durante el tiempo durante que está escuchando el canal, escogerá una de ellas para realizar el registro a través del *Switch* emisor. A su vez, el nodo de servicio registrará el instante en el que se recibe la baliza escogida y en función de la frecuencia de envío de dicha baliza (indicado por el campo *BCN.FRQ* de la trama *BPDU*) el nodo de servicio sabrá el tiempo mínimo que pasará antes de recibir la siguiente baliza del mismo *Switch*. Por ejemplo, si en la última baliza recibida el campo *BCN.FRQ* es igual a 5, esto significa que el *Switch* emisor transmite 1 baliza por cada 32 tramas. Por lo tanto, teniendo en cuenta que la longitud de la trama es de 276 símbolos, pasarán 8832 símbolos (19,784 segundos) antes de que llegue la siguiente baliza. Conociendo esta información, el nodo de servicio inicia un temporizador para llevar el control de las balizas que van llegando correctamente. Cada vez que llega una nueva baliza, este temporizador se reinicia con la información de la última baliza recibida. Así, si un nodo de servicio registrado detecta la falta de $N_{miss-beacon}$ (ver A.2.4 del capítulo A) número de balizas seguidas, asumirá que su *Switch* no está disponible, cambiará al estado *Desconectado* y tratará de registrarse a través de otro *Switch*.

- Proceso *Keep-Alive*. El proceso *Keep-Alive*, tal y como se ha descrito en el apartado 2.2.3.6.8 del capítulo 2 es un mecanismo de la capa MAC que se utiliza para saber cuándo un nodo de servicio ha abandonado la subred. Para ello, cuando un nodo de servicio se registra en la subred, la llegada del paquete de respuesta *REG_RSP* por parte del nodo base hará que en el nodo de servicio se inicie el temporizador $T_{keepAlive}$ cuya duración se obtendrá del campo *REG.TIME* del mencionado paquete de respuesta. Por cada paquete *ALV_B* que recibe el nodo de servicio, se reiniciará el temporizador utilizando el valor del campo *ALV.TIME*. La llegada de un paquete *PRO_REQ_B* (en respuesta a la solicitud de promoción del nodo) también reiniciará el temporizador $T_{keepAlive}$ con el valor del campo *PRO.TIME*. El criterio seguido por el nodo base para escoger los valores *REG.TIME, ALV.TIME* o *PRO.TIME* no está especificado en el estándar, por lo que cada fabricante escoge el suyo.

En el modelo de simulación desarrollado, la asignación de estos valores se ha realizado mediante un algoritmo basado en el nivel de confianza de los nodos. Tal y como indica el estándar, por cada paquete *ALV_B* o *ALV_S*

recibido por el nodo de servicio o el nodo base, el contador *ALV.RXCNT* será incrementado. En el algoritmo implementado, cuando un nodo de servicio se registra en el nodo base, el valor del campo *REG.TIME* del paquete *REG_RSP* será pequeño (por ejemplo 0 que equivale a 32 segundos), y la frecuencia de envío de paquetes de *ALV_B* será muy alta (por ejemplo, se enviará 1 paquete cada 20 segundos). Así, por cada 3 intercambios de paquetes *ALV* llevados a cabo con éxito entre el nodo base y un nodo de servicio, el nivel de confianza que tiene el nodo base sobre dicho nodo de servicio aumentará, por lo que el valor del campo *ALV.TIME* se irá incrementando (hasta llegar al valor 7 que equivale a 68,3 minutos). A su vez, la frecuencia de envío de paquetes *ALV_B* por parte del nodo base a dicho nodo se reducirá.

- Retransmisión de paquetes de control. El mecanismo de retransmisión se ha implementado tal y como se ha explicado en el apartado 2.2.3.7 del capítulo 2. Para ello cada vez que se ha realizado una transacción MAC se han programado los temporizadores y contadores necesarios tal y como indica el estándar.

4.2.5.3 Capa de convergencia (CL)

La capa de convergencia clasifica el tráfico asociándolo con su conexión MAC correcta. En el modelo de simulación desarrollado, se ha implementado la subcapa de convergencia *NULL* que proporciona una ruta directa desde la capa MAC a las superiores.

4.2.5.4 Capa de aplicación (APP)

En esta capa se ha implementado la aplicación de actualización *firmware* descrita en el apartado 2.2.4.2 del capítulo 2. Los detalles de dicha implementación se incluyen más adelante en el apartado 4.3 de este capítulo.

4.2.6 Network Manager

El módulo *Network Manager*, es un módulo que no existe en las redes reales pero que lleva a cabo funciones muy importantes que de otro modo sería más complicado llevarlas a cabo en el entorno de simulación.

Por un lado, los nodos de la subred implementan el mecanismo CSMA/CA para evitar colisiones de paquetes que se producen cuando uno o varios nodos intentan acceder al canal de forma simultánea. En dicho mecanismo, tal y como se explica en el apartado 2.2.3.4.4 del capítulo 2, un nodo que está a punto de transmitir un paquete realiza una comprobación de canal (una o más veces en

función de la prioridad del paquete a transmitir) para saber si está libre. En las redes reales, para que un nodo pueda saber si se está transmitiendo algo por el canal en ese momento, tratará de detectar la presencia de una portadora. Así, en caso de que el nodo detecte una portadora, entenderá que el canal está ocupado. Para poder realizar está comprobación en el entorno de simulación, se ha creado el módulo *Network Manager* gracias al cual, mediante una simple consulta, los nodos pueden conocer si el canal está ocupado en ese momento.

Por otro lado, como en el entorno de simulación no existe un medio físico real, se ha desarrollado un algoritmo que simplifica la reproducción de colisiones y la detección de paquetes corruptos por parte de los nodos.

A continuación, se explicarán el algoritmo de detección de canal y el de reproducción de colisiones.

4.2.6.1 Detección de canal

Cuando un nodo ejecuta el mecanismo CSMA/CA de acceso al medio y se dispone a comprobar si el canal está libre o ocupado, realiza una consulta al módulo *Network Manager*. Es importante destacar, que esta consulta se hace a través de un mecanismo de tipo "puerta trasera" que ofrece el simulador OMNeT++, por lo que no afectará al resultado de las simulaciones.

Cada vez que un nodo envíe un paquete a la línea, el *Network Manager* recibirá una notificación y éste almacenará la información del paquete en un vector durante el tiempo que dure su transmisión. Para ello, cuando el *Network Manager* reciba la notificación del envío de un paquete, calculará la duración de la transmisión del paquete y programará un evento para que pasado ese tiempo se elimine la información del paquete del vector donde previamente se ha almacenado. El cálculo de la duración de la transmisión del paquete se explicará más adelante en el apartado 4.2.7.

Por lo tanto, cuando un nodo quiera transmitir un paquete le preguntará al módulo *Network Manager* si el canal esta libre para transmitir. El *Network Manager*, por su parte, mirará si en el vector donde se almacena la información de los paquetes en transmisión hay algún paquete. En caso de que el vector esté vacío, el módulo *Network Manager* notificará al nodo que el canal está libre. Sin embargo, si en el vector hay algún paquete, se comprobará el nivel lógico en la jerarquía de los nodos que han enviado cada uno de los paquetes del vector y se comparará con el nivel lógico del nodo que desea transmitir. A su vez, se calculará la diferencia de niveles máxima que debe haber entre dos nodos para que puedan escucharse entre sí. Esta diferencia de niveles máxima es el resultado del cálculo de un valor aleatorio basado en el parámetro *collisionDomain*, tal y como se ha explicado en el apartado 4.2.4 de este capítulo. Así, si la diferencia de niveles entre los nodos origen de los paquetes almacenados en el vector y el nodo que se dispone a transmitir es menor o igual al número aleatorio cal-

culado, el *Network Manager* determinará que el canal está ocupado (siempre y cuando todos los nodos se encuentren en el dominio del mismo transformador, en caso de que hubiera más de uno en la subred). En cambio, si la diferencia de niveles entre los nodos es mayor que el valor aleatorio calculado, el *Network Manager* informará al nodo de que el canal está libre.

4.2.6.2 Algoritmo de reproducción de colisiones

Cuando un nodo está a punto de transmitir un paquete, le pregunta al *Network Manager* si el canal está libre para transmitir. Si es así, el nodo comenzará a transmitir el paquete y enviará una notificación al *Network Manager* para que éste almacene la información del paquete en un vector de paquetes en transmisión. Dependiendo del tipo de paquete, éste tendrá un tamaño diferente por lo que en función del esquema de modulación empleado en la transmisión, la duración de la transmisión del paquete será distinta. Por lo tanto, el módulo *Network Manager* calculará la duración de la transmisión y después de que transcurra este tiempo eliminará la información del paquete del vector. Si durante el tiempo en el cual el paquete se encuentra almacenado en el vector otro nodo transmite un paquete y se lo notifica al *Network Manager*, este último determinará si ha ocurrido una colisión entre los paquetes enviados en función de la diferencia de niveles existente entre los nodos transmisores. La diferencia de niveles máxima para que exista una colisión vendrá determinada por un número aleatorio calculado a partir del parámetro *collisionDomain*.

En caso de que ocurra una colisión, el *Network Manager* almacenará el identificador de los paquetes que han colisionado en un vector de paquetes corruptos. Así, cuando un nodo reciba un paquete realizará una consulta al *Network Manager* para saber si dicho paquete se ha recibido correctamente o no. Si el identificador del paquete se encuentra en el vector de paquetes corruptos, el nodo receptor descartará el paquete, mientras que si no lo está, el paquete será transmitido a las capas superiores del nodo.

4.2.7 Cálculo de la duración de la transmisión de paquetes

En el presente subapartado se explicará el procedimiento seguido para el cálculo de la duración las tramas a nivel de capa física.

Por ejemplo, para calcular la duración del paquete *ALV* empleado en el proceso *Keep-Alive*, al tratarse de un paquete de tipo GPDU (ver sección 2.2.3.5.1 del capítulo 2) su estructura se compone de una cabecera MAC genérica (3 $bytes$) seguido de uno o más paquetes MAC y un CRC (4 $bytes$) al final (figura 4.22).

A su vez, un paquete MAC se compone de una cabecera de 6 $bytes$ y un *payload* en el que se incluirá la información del paquete de control MAC a transmitir. En el ejemplo escogido, al tratarse de un paquete de control de tipo *ALV*, el ta-

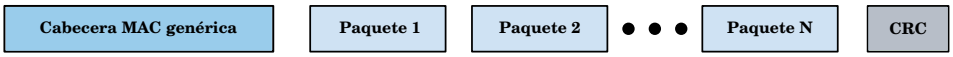


Figura 4.22: Estructura general de la trama GPDU

maño del *payload* del paquete MAC es de 3 $bytes$, por lo que el tamaño total de la trama GPDU es de 16 $bytes$ (figura 4.24).

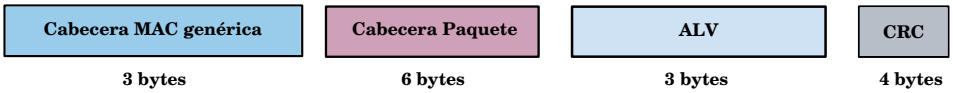


Figura 4.23: Estructura de la trama GPDU del paquete de control *ALV*

La trama MPDU (de tipo GPDU) generada se transmite a la capa física donde se generará la trama *PHY* (trama PPDU + preámbulo). La trama *PPDU* consta de una cabecera de capa física (PHY) cuya duración es de dos símbolos OFDM y un *payload*. El *payload* se compone de una o varias *MAC Service Data Unit (MSDU)*, bits de *flushing* en caso de que el mecanismo FEC esté activado (8 bits) y un campo de *padding* para asegurarnos de que el número de bits que componen el *payload* llena un número entero de símbolos OFDM. Es importante mencionar que los 3 bytes de la cabecera genérica MAC y los primeros 4 bytes de la cabecera del primer paquete se incluyen en la cabecera de capa PHY.

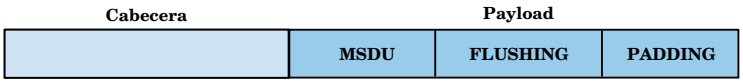


Figura 4.24: Estructura de la cabecera y *payload* del la trama PPDU

La cabecera de capa PHY se transmite siempre empleado la modulación DBPSK y con el mecanismo FEC activado. En cambio, el *payload* puede ser modulado empleando DBPSK, DQPSK, D8PSK con la activación opcional del FEC, dependiendo de la configuración realizada por la capa MAC. Por lo tanto, en función de la modulación empleada, cada símbolo transportará un número concreto de bits de información. La tabla 4.2 muestra en función de la modulación y codificación escogida, el número de bits de información que se pueden transmitir por cada símbolo OFDM.

Tabla 4.2: Número de bits de información por símbolo OFDM

	DBPSK		**DQPSK**		**D8PSK**	
Cód. conv. (1/2)	On	Off	On	Off	On	Off
Bits de información por símbolo OFDM	48	96	96	192	144	288

Por último, a la trama PPDU se le añade un preámbulo de una duración de 2,048 *ms*. El apartado A.1.1 del apéndice A muestra con más detalle la estructura de la trama a nivel de capa PHY.

Por lo tanto, si el paquete *ALV* se transmite con el esquema de modulación más robusto (DBPSK) y con el FEC activado su duración será de 11,008 ms y se calcula como sigue.

El tamaño de la trama MPDU en bytes se muestra en la ecuación 4.3.

$$\text{MPDU (bytes)} = \underbrace{3 \text{ bytes}}_{\text{Cabecera MAC genérica}} + \underbrace{6 \text{ bytes}}_{\text{Cabecera paquete}} + \underbrace{3 \text{ bytes}}_{\text{Payload paquete}} + \underbrace{4 \text{ bytes}}_{\text{CRC}} = 16 \text{ bytes} \tag{4.3}$$

El MPDU se transmite a la capa PHY y los 3 bytes de la cabecera MAC genérica y los primeros 4 bytes de la cabecera del primer paquete MAC se incluyen en la cabecera PHY. El resto de los bytes de la trama MPDU se añaden como MSDU en la trama PPDU. El tamaño de dicho campo se muestra en las ecuaciones 4.4 y 4.5. Para formar el *payload* PPDU, teniendo en cuenta que el código FEC está activado, es necesario añadir los bits de *flushing* tal y como se muestra en la ecuación 4.6. El número de símbolos OFDM necesarios para enviar los bits del campo MSDU y *flushing* empleando la modulación DBPSK con el FEC activado se calcula en la ecuación 4.7. Como el número de bits no es suficiente para llenar un número entero de símbolos OFDM se añade un campo de *padding*. El cálculo del número de bits de dicho campo se muestra en la ecuación 4.8. Por último, la ecuación 4.9 muestra la duración del *payload* de la trama PPDU mientras que la ecuación 4.10 muestra la duración total de la trama PHY.

$$\text{MSDU (bytes)} = \underbrace{16 \text{ bytes}}_{\text{MPDU}} - \underbrace{3 \text{ bytes}}_{\text{Cabecera MAC genérica}} - \underbrace{4 \text{ bytes}}_{\text{Cabecera paquete}} = 9 \text{ bytes} \tag{4.4}$$

$$\text{MSDU (bits)} = 9 \text{ bytes} \times \frac{8 \text{ bits}}{1 \text{ byte}} = 72 \text{ bits} \tag{4.5}$$

$$\text{MSDU} + \textit{flushing} \text{ (bits)} = \underbrace{72 \text{ bits}}_{\text{MSDU}} + \underbrace{8 \text{ bits}}_{\text{flushing}} = 80 \textit{ bits} \tag{4.6}$$

$$\text{MSDU} + \textit{flushing} \text{ (símbolos)} = 80 \times \frac{1 \text{ símbolo}}{48 \text{ bits}} = 1,67 \approx 2 \text{ símbolos} \tag{4.7}$$

$$\textit{Padding} \text{ (bits)} = 2 \text{ símbolos} \times \frac{48 \text{ bits}}{1 \text{ símbolo}} - 80 \text{ bits} = 16 \text{ bits} \tag{4.8}$$

$$\text{Payload PPDU (ms)} = 2 \text{ símbolos} \times \frac{2,24 \text{ ms}}{\text{símbolo}} = 4,48 \text{ ms} \tag{4.9}$$

$$\text{Trama PHY (ms)} = \underbrace{2,048\ \text{ms}}_{\text{Preámbulo}} + \underbrace{\underbrace{4,48\ \text{ms}}_{\text{Cabecera}} + \underbrace{4,48\ \text{ms}}_{\text{Payload}}}_{\text{PPDU}} = 11,008\ \text{ms} \tag{4.10}$$

4.2.8 Señales

Tal y como se ha descrito en el apartado 4.2.5 de este capítulo, para reproducir el envío de tramas *BPDU* o balizas, se ha empleado el mecanismo de señales que ofrece OMNeT++.

Las señales son emitidas por componentes (módulos y canales) y se propagan en la jerarquía hasta la raíz. En cualquier nivel, se pueden registrar *listeners* (objetos *callback*) a los que les llegará una notificación cada vez que se emita cualquier señal.

Estas señales se identifican mediante nombres, aunque el simulador internamente por cuestiones de eficiencia les asigna identificadores numéricos de forma dinámica. El mapeo de nombres de señal a identificadores de señal es global, por lo que todos los módulos y canales que quieran resolver el nombre de una señal particular obtendrán el mismo identificador numérico de la señal.

Los *listeners* se pueden suscribir a cualquier tipo de señal (mediante nombre o ID) independientemente de su fuente. Por ejemplo, dos módulos distintos que no están relacionados entre sí, llamados módulo A y B, emiten una señal (cada uno con un mismo nombre) que será recibida por otros módulos en la simulación. El método *listener* de estos últimos podrá comprobar el origen de la señal si se quiere distinguir entre la señal enviada por A y la señal enviada por B. En el caso concreto del envío de balizas, pongamos que tenemos dos *Switches* (además del nodo base) en la subred que emiten señales con el nombre “baliza” y que cada una se emite en su *slot* correspondiente. En el modelo de simulación desarrollado, todos los nodos de la subred estarán suscritos a señales de tipo “baliza”, por lo que cuando reciban este tipo de señal, en su método *listener* tendrán que comprobar el origen de dicha señal. Si el *Switch* que la ha emitido se encuentra entre la lista de padres más comunes de dicho nodo de servicio (indicado por el fichero XML con la topología lógica más común de la subred), el nodo de servicio tratará de registrarse a través de dicho *Switch* (en caso de que esté en estado *Desconectado*). En cambio, si el *Switch* emisor de la baliza no se encuentra entre los padres, el nodo de servicio ignorará la señal.

Por otro lado, las señales también se utilizan para grabar datos del modelo que permiten extraer información estadística. De este modo, a la hora de ejecutar las simulaciones se pueden recoger datos muy interesantes como el número de paquetes enviados, el número de colisiones ocurridas, tiempo total que dura un proceso, el tiempo en el que un nodo de servicio ha estado en estado *Desconectado*, *Terminal* o *Switch*, etc.

Existen dos formatos para grabar datos de resultados de simulación que son los vectores y escalares de salida. Los vectores se utilizan para registrar datos como paquetes perdidos, cambios de estado de módulos, paquetes enviados, tiempos de espera, retardos, etc., es decir, todo lo que es útil para obtener una imagen completa de lo ocurrido en el modelo durante la ejecución de la simulación. Los escalares en cambio son los resultados de resumen calculados durante la simulación y escritos cuando la simulación termina. Un resultado escalar puede ser un número (entero o real), o puede ser un resumen estadístico compuesto por varios campos, como el recuento, media, desviación estándar, suma, mínimo, máximo, etc., y opcionalmente los datos de un histograma. En el modelo de simulación desarrollado, los escalares se han empleado por ejemplo, para grabar los instantes de inicio y fin del proceso de actualización *firmware*.

4.3 Validación del modelo de simulación

Para llevar a cabo el proceso de validación del modelo de simulación desarrollado, se ha implementado y evaluado una estrategia de actualización *firmware* tanto en el entorno de simulación como en entornos reales. Los resultados de los experimentos realizados sobre los escenarios reales han sido facilitados por Iberdrola. El objetivo de la validación es verificar el correcto funcionamiento del modelo desarrollado de modo que pueda ser utilizado como escenario de pruebas en el estudio de estrategias de actualización *firmware* presentado en este trabajo de investigación.

4.3.1 Estrategia de actualización *firmware* implementada

Como ya se ha adelantado, para llevar a cabo la validación del modelo de simulación, se ha implementado la aplicación de actualización *firmware*. En concreto, se ha implementado una estrategia de actualización que actualmente ya se utiliza en redes reales y que cuyo funcionamiento se muestra en la figura 4.25. Para comprender los detalles de la estrategia y poder replicar su implementación en el modelo de simulación, se han empleado los ficheros de trazas generados durante la ejecución de la estrategia sobre las redes reales escogidas y que han sido facilitados por Iberdrola. Los parámetros más importantes empleados en dicha implementación se incluyen en la tabla 4.3.

Por lo tanto, la estrategia de actualización implementada funciona como sigue. En primer lugar, el nodo base define un único grupo *multicast* en el que se incluye a todos los nodos de un mismo fabricante, ya que estos ejecutan el mismo *firmware*. Después, el nodo base establece una conexión con cada uno de los nodos que formarán parte dicho grupo, les pide que se unan al grupo *multicast* y les indica que el proceso de actualización de *firmware* está a punto de comenzar. Si alguno de estos nodos de servicio se encuentra en el estado

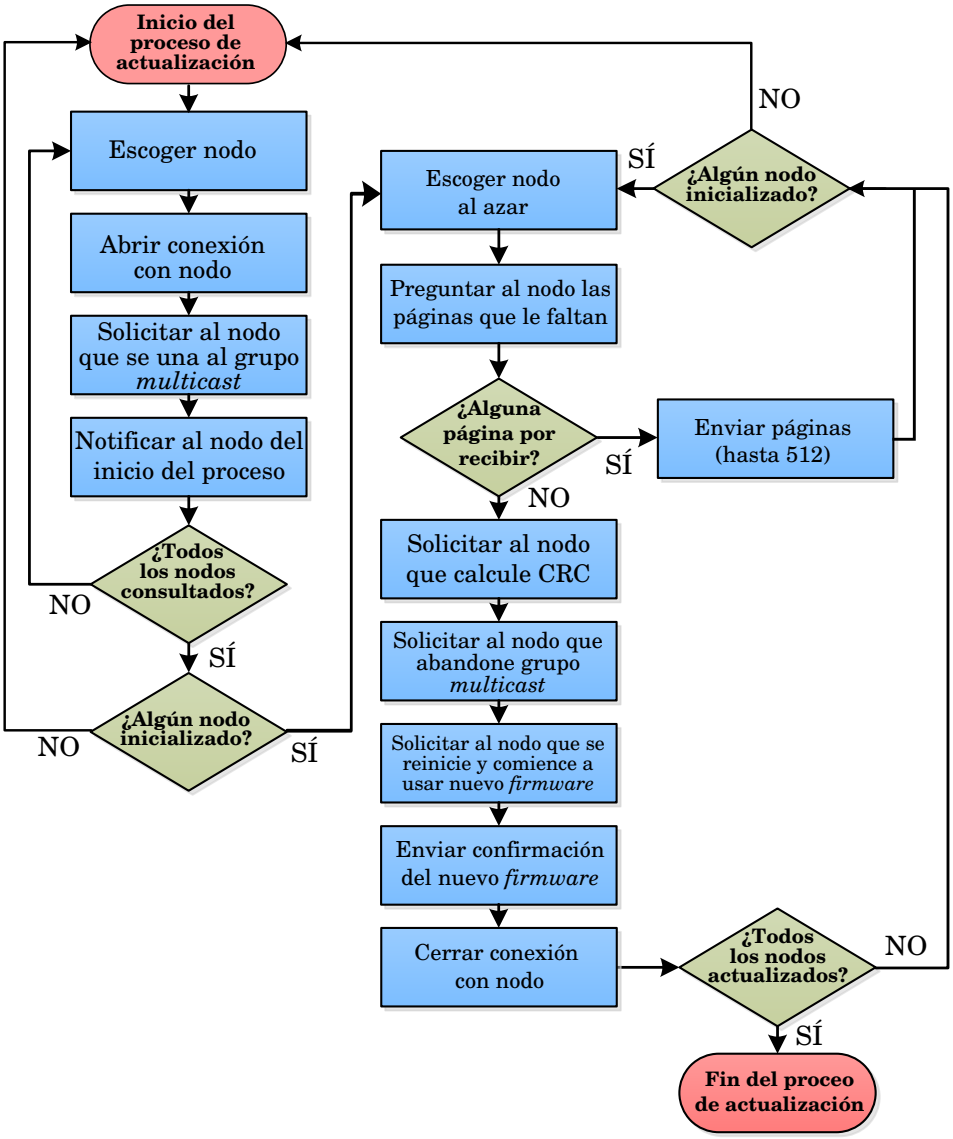


Figura 4.25: Estrategia de actualización *firmware* con activación inmediata

Tabla 4.3: Parámetros de simulación

Parámetro	Valor
Tamaño del *firmware*	98.432 ≈ 100 kBytes
Tamaño de página	64 Bytes
Núm. de páginas	1538 páginas
Máx. número de páginas a enviar	512 páginas
Espaciado entre paquetes	600 milisegundos
Restart Timer	0 segundos
Safety Timer	32400 segundos
Esquema de modulación	DBPSK con FEC

Desconectado y por lo tanto no puede ser inicializado, el nodo base continúa con el resto de los nodos y deja la actualización de los nodos en este estado para la siguiente ronda de actualización.

Después de haber inicializado el mayor número de nodos de servicio posible, el nodo base escoge un nodo de servicio al azar al que solicitará la lista de páginas del *firmware* que le quedan por recibir. En los ejemplos del proceso de actualización mostrados en el estándar PRIME (ver sección 2.2.4.2 del capítulo 2), después de inicializar el nodo de servicio, el nodo base comienza a enviar las páginas del *firmware* directamente. Sin embargo, como los nodos de servicio puede que hayan guardado páginas del *firmware* en un proceso de actualización anterior sin finalizar, en esta estrategia, antes de comenzar con el envío de páginas, el nodo base solicitará a los nodos de servicio que le indiquen las páginas que les faltan por recibir. Por lo tanto, el nodo base escoge un nodo aleatorio al que envía dicha solicitud y comienza con el envío de las páginas indicadas por el nodo de servicio, enviando hasta un máximo de 512 páginas con un espaciado de 600 milisegundos. Después de enviar las páginas, escogerá otro nodo de servicio al azar sin actualizar y le solicitará la lista de páginas que le quedan aún por recibir. Este proceso se repite hasta que un nodo de servicio indica al nodo base que ya no le queda ninguna página por recibir. Cuando esto ocurre, el nodo base solicita al nodo de servicio que compruebe la integridad de la imagen del *firmware* y en caso de éxito, el nodo base solicitará al nodo de servicio que abandone el grupo *multicast*. Después, el nodo base le enviará un paquete de ejecución del *firmware* para que el nodo de servicio se reinicie y comience a utilizarlo. Finalmente, para terminar la actualización de este nodo de servicio, el nodo base le deberá enviar un paquete de confirmación del nuevo *firmware* antes de que el temporizador *Safety Timer* llegue a su fin. De lo contrario, el nodo de servicio rechazará la nueva imagen y volverá a la versión antigua del *firmware*.

Después de haber finalizado la actualización de este nodo, el nodo base cierra la conexión con él y continúa con la actualización del resto de nodos de servicio de la red. Para ello, el nodo base escoge otro nodo sin actualizar al azar y le pregunta por las páginas que le quedan por recibir. Este proceso se repite hasta que todos los nodos del grupo *multicast* hayan sido actualizados o hasta que no queden nodos inicializados a los que preguntar por sus páginas pendientes. En el segundo caso, el proceso de actualización volverá al punto de partida con el fin de actualizar el resto de nodos de servicio del grupo.

4.3.2 Escenarios de prueba

El modelo de simulación desarrollado se ha validado empleando los datos de tres escenarios reales distintos. Los detalles de los escenarios empleados se incluyen a continuación.

- **Escenario 1**

  Esta red está compuesta por el nodo base y 68 nodos de servicio, de los cuales 58 han de ser actualizados. La figura 4.26 muestra la topología lógica más común obtenida a partir de los datos proporcionados por el nodo base de la red real. Como se puede observar la red se compone de 4 niveles de jerarquía.

- **Escenario 2**

  El segundo escenario esta compuesto por el nodo base y 45 nodos de servicio de los cuales 33 son para actualizar. La figura 4.27 muestra la topología más común de esta red. En este caso la red se compone de 4 niveles de jerarquía.

- **Escenario 3**

  El tercer escenario se compone de un nodo base, 69 nodos de servicio y 58 nodos de servicio a actualizar. En este caso, la figura 4.28 muestra la topología más común para esta red, la cual se compone de 2 niveles de jerarquía.

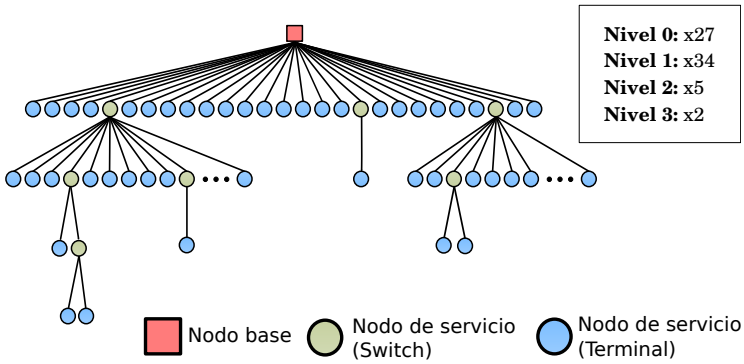


Figura 4.26: Topología lógica más común del escenario 1

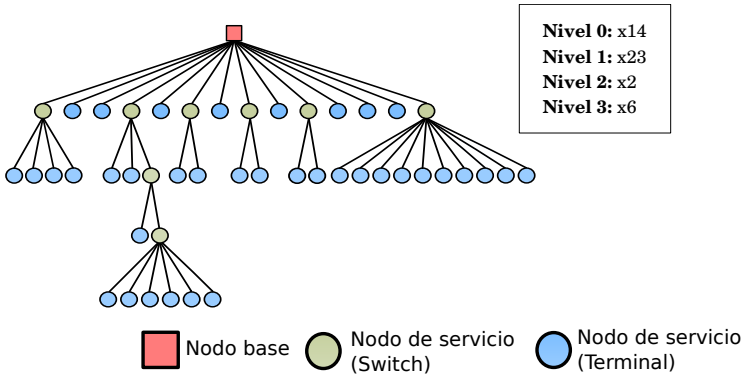


Figura 4.27: Topología lógica más común del escenario 2

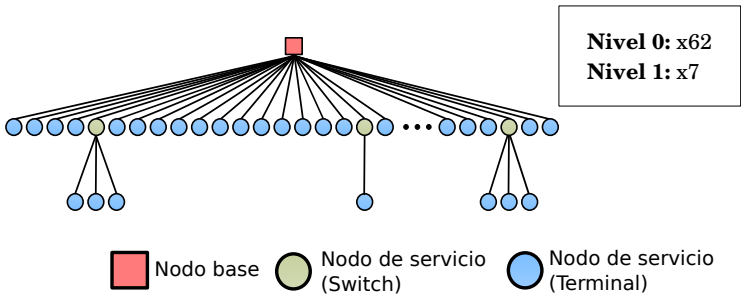


Figura 4.28: Topología lógica más común del escenario 3

4.3.3 Análisis e interpretación de resultados

Los resultados de los experimentos realizados en la validación se muestran en la tabla 4.4. En dicha tabla se muestra la duración del proceso de actualización llevado a cabo en cada uno de los escenarios, tanto en el entorno real como en el simulado. Para conocer si los resultados de tiempo de actualización

Tabla 4.4: Resultados de los experimentos

Primer escenario			
Tipo de exp.	**Tiempo medio**	**Desv. estándar**	**Tamaño de muestra**
Real	2 h 54 min	0.79	6
Simulado	2 h 42 min	0.41	10
Segundo escenario			
Tipo de exp.	**Tiempo medio**	**Desv. estándar**	**Tamaño muestra**
Real	3 h 23 min	0.69	3
Simulado	3 h 22 min	0.22	10
Tercer escenario			
Tipo de exp.	**Tiempo medio**	**Desv. estándar**	**Tamaño muestra**
Real	1 h 19 min	0.39	6
Simulado	1 h 23 min	0.15	10

obtenidos dependen del tipo de experimento realizado (en el entorno real o de simulación) se analizará la relación que existe entre la variable cualitativa "tipo de experimento" y la cuantitativa "tiempo de actualización total de la subred". De este modo, si se demuestra que los resultados de tiempo de actualización son independientes del tipo de prueba realizado, podremos concluir que el modelo de simulación reproduce correctamente el comportamiento de subredes PRIME y por lo tanto, quedaría validado.

Debido al alto coste y complejidad de las pruebas de campo, el número de repeticiones por cada experimento en el escenario real es pequeño, por lo que para realizar el análisis mencionado y verificar así que los resultados de las simulaciones se asemejan a los resultados de campo, se ha llevado a cabo la prueba *t de student* para dos muestras [SC89] mediante el *software* estadístico R [R C13].

El primer paso a la hora de realizar la prueba, es el de establecer una hipótesis nula y una alternativa. La hipótesis nula indica que el tiempo de actualización no depende del tipo de experimento realizado (en un entorno real o de simulación), mientras que la hipótesis alternativa sostiene que el tiempo de actualización sí depende del tipo experimento. Para estudiar si existe relación entre estas dos variables, es decir, entre el tiempo de actualización (variable cuantitativa) y el tipo de experimento (variable cualitativa), se realiza un contraste de comparación de la media del tiempo de actualización de los dos grupos de resultados; es decir, de los resultados en el entorno real y en el entorno de simulación. Si el tiempo medio de ambos grupos es el mismo, significa que el tiempo de actualización no depende del tipo de experimento realizado. Por lo

Figura 4.29: *Box-plot* para conocer si los datos siguen una distribución normal

tanto, la hipótesis nula, tal y como se muestra en la ecuación (4.11), es que la diferencia de medias entre las dos muestras es 0, mientras que la hipótesis alternativa, mostrada en (4.12), establece que la diferencia entre las muestras es distinta de 0. La variable μ_1 es la media de la población de la muestra de resultados del entorno real, mientras que μ_2 es la media de la población de la muestra de los resultados de las simulaciones.

$$H_0 : \mu_1 = \mu_2 \tag{4.11}$$

$$H_a : \mu_1 \neq \mu_2 \tag{4.12}$$

Antes de realizar el análisis es importante saber si los datos con los que se va a trabajar se ajustan o no a una distribución normal, ya que el modo de trabajar será diferente. Para ello, mediante la representación de los datos en un *box-plot* o diagrama de cajas se comprueba dónde queda la línea que representa la mediana. Si la mediana está centrada con respecto al primer y tercer cuartil, esto indica que la distribución es simétrica y se corresponde con una distribución normal. En este caso, tal y como se muestra en la figura 4.29 los datos a estudiar siguen una distribución normal.

Por otro lado, teniendo en cuenta que estamos trabajando con muestras que son independientes es necesario conocer si dichas muestras se corresponden con poblaciones que tienen la misma varianza. Así, para poblaciones con varianza desigual se llevará a cabo la prueba *t de Welch* [RP61], mientras que si las varianzas resultan iguales se realizará una prueba *t de student* estándar.

Para determinar cuál de las dos pruebas se debe utilizar, se ha llevado a cabo la prueba de *Levene* [Fox15] en cada caso. La prueba de *Levene* es una prueba de estadística inferencial utilizada para evaluar la igualdad de las varianzas de una variable calculada para dos o más muestras.

En el caso que nos ocupa, la prueba de *Levene* indica que en los tres casos las muestras son homocedásticas, es decir, la varianza de las muestras es igual, por lo que se ha llevado a cabo una prueba *t de student* estándar en los tres casos.

Como resultado de la prueba, R nos devuelve el *p*-valor y el *t*-valor. Estos valores sirven de ayuda a la hora de decidir si rechazar o no la hipótesis nula. Hay que tener en cuenta que solamente es necesario tener uno de los dos valores para tomar esta decisión. En caso de utilizar el *p*-valor, tal y como se muestra en la ecuación 4.14, este valor se compara con el nivel de significación estadística (α). El valor de significación estadística se define como la probabilidad de tomar la decisión de rechazar la hipótesis nula, cuando esta es verdadera; es decir, la probabilidad que tenemos de equivocarnos al rechazar la hipótesis nula. En el caso que nos ocupa, se ha escogido un valor α del 5 % (nivel de confianza de 95 %). Si al realizar la prueba el valor *p*-valor devuelto por R es menor que 0,05 asumimos que el riesgo es pequeño a la hora de rechazar la hipótesis nula, por lo que deduciríamos que las variables (cualitativa y cuantitativa) están relacionadas y por lo tanto, en este caso los resultados de tiempo de actualización sí que dependerían del tipo de prueba realizado.

Para poder tomar esta misma decisión mediante el *t*-valor, el resultado de la prueba proporcionado por R se comparará con un valor *T* crítico que puede ser consultado en una tabla como la que se muestra en la figura 4.31. Si se cumple la condición que se muestra en la ecuación (4.15), también podríamos rechazar la hipótesis nula. Para escoger el valor *T* crítico de la tabla es necesario conocer el valor de los grados de libertad (*gl*). En este caso, teniendo en cuenta que la prueba de *Levene* indica que las muestras son homocedásticas, los grados de libertad se calculan aplicando la siguiente fórmula:

$$\text{Grados de libertad (gl)} = n1 + n2 - 2 \tag{4.13}$$

donde $n1$ y $n2$ son el tamaño de la muestra de las pruebas en el entorno real y en el de simulación respectivamente. Por otro lado, debido a que se va a realizar un contraste bilateral de hipótesis, la región de rechazo se sitúa en los dos extremos (colas) y por lo tanto para obtener el valor *T* de la tabla nos tendremos que fijar en el valor $\alpha/2$ (figura 4.30).

$$p\text{-}valor < \alpha \tag{4.14}$$

$$|t\text{-}valor| \geq T_{\alpha/2,gl} \tag{4.15}$$

La tabla 4.5 muestra los resultados de la prueba *t de student* realizada mediante R y como se puede observar, en todos los casos el *t*-valor es más pequeño que el valor *T* crítico ($T_{\alpha/2,gl}$), mientras que su *p*-valor es mayor que el nivel de significación escogido en todos los casos.

Por lo tanto, debido a que la hipótesis nula no puede ser rechazada, ya que la probabilidad de cometer una equivocación es muy alta, se puede concluir

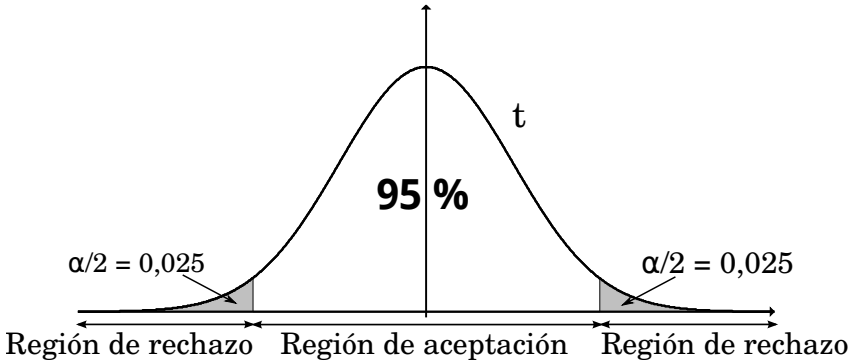


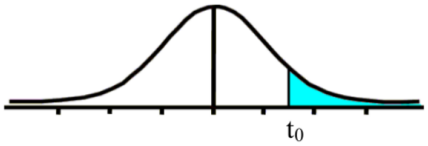
Figura 4.30: Contraste bilateral de hipótesis

Tabla 4.5: Resultados de la prueba *t de student* para dos muestras independientes obtenidos mediante R

Escenario	t-valor	p-valor	α	df	$T_{\alpha/2,df}$
Escenario 1	0.6658	0.5163	0.05	14	2.1448
Escenario 2	0.0366	0.9714	0.05	11	2.2010
Escenario 3	-0.429	0.6742	0.05	14	2.1448

que las medias poblacionales de las dos muestras en cada escenario son iguales. En consecuencia, no hay ninguna razón para considerar los resultados de los experimentos dependen del tipo de experimento (real o simulado) por lo que se puede concluir que el modelo de simulación desarrollado es válido.

Tabla t-Student



Grados de libertad	0.25	0.1	0.05	0.025	0.01	0.005
1	1.0000	3.0777	6.3137	12.7062	31.8210	63.6559
2	0.8165	1.8856	2.9200	4.3027	6.9645	9.9250
3	0.7649	1.6377	2.3534	3.1824	4.5407	5.8408
4	0.7407	1.5332	2.1318	2.7765	3.7469	4.6041
5	0.7267	1.4759	2.0150	2.5706	3.3649	4.0321
6	0.7176	1.4398	1.9432	2.4469	3.1427	3.7074
7	0.7111	1.4149	1.8946	2.3646	2.9979	3.4995
8	0.7064	1.3968	1.8595	2.3060	2.8965	3.3554
9	0.7027	1.3830	1.8331	2.2622	2.8214	3.2498
10	0.6998	1.3722	1.8125	2.2281	2.7638	3.1693
11	0.6974	1.3634	1.7959	2.2010	2.7181	3.1058
12	0.6955	1.3562	1.7823	2.1788	2.6810	3.0545
13	0.6938	1.3502	1.7709	2.1604	2.6503	3.0123
14	0.6924	1.3450	1.7613	2.1448	2.6245	2.9768
15	0.6912	1.3406	1.7531	2.1315	2.6025	2.9467

Figura 4.31: Tabla de valores *t* críticos para la prueba *t de Student* para dos muestras independientes

Capítulo

5

Estudio de estrategias de actualización *firmware*

En este capítulo se incluye un estudio que consiste en el diseño, implementación y evaluación de distintas estrategias de actualización *firmware*. La sección 5.1 proporciona una visión general de la problemática abordada en este trabajo de investigación. A continuación, en la sección 5.2 se presentan los aspectos de libre implementación del proceso de actualización *firmware* definido en el estándar PRIME. Por su parte, la sección 5.3 describe los criterios seguidos a la hora de diseñar las estrategias de actualización, así como el funcionamiento de las mismas en detalle. Por último, en la sección 5.4 se lleva a cabo la evaluación de las estrategias de actualización *firmware* propuestas en este trabajo de investigación.

5.1 Introducción

Como ya se ha adelantado, con la aparición de las *Smart Grids* surgirán nuevos retos tecnológicos que exigirán la mejora de las prestaciones de tecnologías como PRIME. Este trabajo de investigación pone su atención en la disponibilidad de las subredes PRIME como recurso crítico de las *Smart Grids*. La disponibilidad PRIME de un nodo de servicio se define como la cantidad de tiempo durante el cual un nodo puede comunicarse con otros nodos de la red (se encuentra en un estado disponible, es decir, en estado *Terminal* o *Switch*) comparándolo con el tiempo total de observación, mientras que la disponibilidad de la subred corresponde a la media de las disponibilidades de todos los elementos de la subred [SBA+13] (figura 5.1).

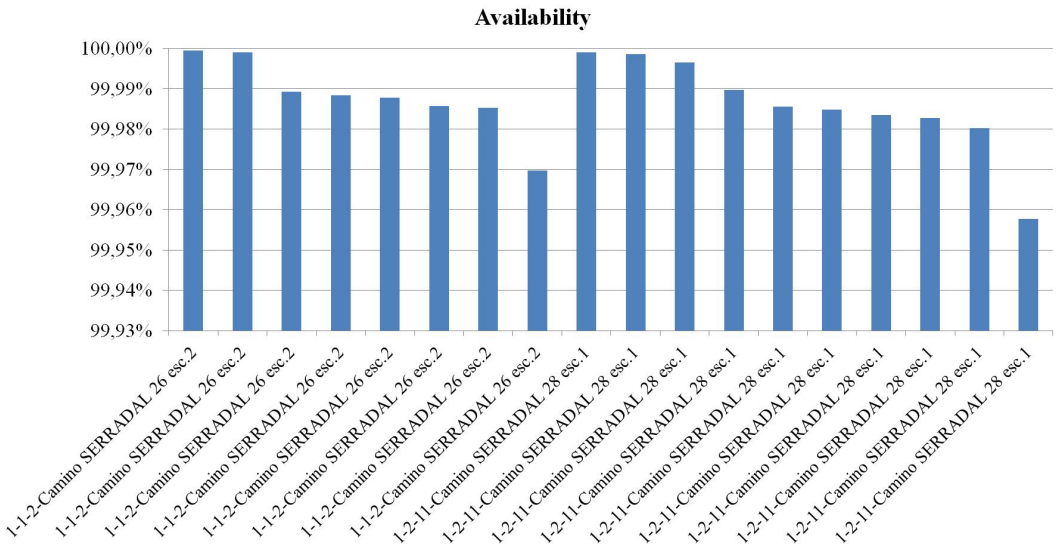


Figura 5.1: Ejemplo de concepto de disponibilidad en una subred PRIME de una prueba de campo real. Cada barra representa la disponibilidad de un nodo de servicio [SBA+13].

La aplicación de actualización *firmware* de los nodos de servicio que componen una subred PRIME es especialmente relevante en términos de disponibilidad de subred ya que cuando los nodos de servicio reciben la imagen del *firmware*, se requiere el reinicio de cada uno de ellos de modo que puedan comenzar a utilizar la nueva versión. Este reinicio supone un empeoramiento de la disponibilidad de dicho nodo así como el de todos sus elementos dependientes, lo que se traduce, a su vez, en un empeoramiento de la disponibilidad de la subred.

Actualmente, no existe un consenso acerca de cuál es la forma más eficiente de implementar dicha aplicación, ya sea en términos de disponibilidad de red, o en otros términos como el de tiempo de actualización. Esto se debe principalmente a que, a pesar de que PRIME define la aplicación de actualización en gran medida, algunos aspectos clave de la aplicación son de libre implementación por lo que cada fabricante la desarrolla como considere oportuno.

Así, en las siguientes secciones, se incluye la principal aportación científica de este trabajo de investigación que consiste en un estudio que incluye el diseño, implementación y evaluación de distintas estrategias de actualización *firmware* con el objetivo de mejorar la disponibilidad sin aumentar el tiempo total del proceso.

5.2 Proceso de actualización *firmware* de PRIME

Como ya se ha comentado, actualmente, no existe un consenso acerca de cuál es la forma más eficiente de implementar la aplicación de actualización *firmware* de los equipos de una red PRIME.

Además, tal y como se ha observado en el capítulo 3, el problema de la disponibilidad de las subredes PRIME durante el proceso de actualización *firmware* nunca antes ha sido abordado en la literatura. Tampoco se han encontrado trabajos que realicen el estudio de distintas estrategias de actualización desde otros puntos de vista distintos al de mejorar la disponibilidad de la red. Por lo tanto, actualmente, el único documento del cual partir es el propio estándar de PRIME. A continuación, se enumeran distintos los aspectos de libre implementación que se han detectado y que han sido tomados en cuenta a la hora de realizar el diseño de las distintas estrategias de actualización *firmware* presentadas en este trabajo.

- **Modo *unicast* o *multicast***. La aplicación de actualización *firmware* puede trabajar en modo *unicast* o *multicast*. En caso de que los nodos de servicio soporten el modo *multicast*, a pesar de que el estándar no las especifica, existen muchas formas de definir los grupos (en función de los niveles lógicos de la jerarquía, ramas, etc.).

- **Tamaño de páginas del *firmware***. Los paquetes de datos en los que se enviarán las páginas del *firmware* podrán ser configurados para que tengan un tamaño u otro en función de las condiciones de canal, pero este tamaño nunca podrá ser cambiado durante el proceso de descarga de la imagen. El tamaño de las páginas podrá ser de 32, 64, 128 o 192 bytes y en función de este parámetro el número de páginas a enviar variará.

- **Espaciado temporal del envío de páginas**. La separación temporal con la que el nodo base transmite las páginas del *firmware* también es de libre configuración.

- **Número de páginas de *firmware* seguidas a enviar**. En los ejemplos del proceso de actualización *firmware* mostrados por el estándar, justo después de que el nodo base envía el paquete de inicialización a un nodo de servicio y recibe su respuesta, comienza con el envío de las páginas del *firmware*. En lugar de proceder de este modo, (enviando las páginas directamente) el nodo base tiene la opción preguntar primero por las páginas del *firmware* que le quedan por recibir a un nodo de servicio, y enviar un número concreto de páginas antes de volver a preguntar a otro y enviar más páginas, ya que es posible que los nodos de servicio ya hayan recibido páginas en algún proceso de actualización anterior sin finalizar. El número de páginas a enviar cada vez que el nodo base realiza dicha consulta a

un nodo de servicio y el modo en el que el nodo base escoge a qué nodo de servicio preguntar no se concreta en el estándar.

- **Reinicio de nodos al mismo tiempo o en varios pasos**. Una vez que un nodo de servicio haya recibido la imagen del nuevo *firmware* al completo, solamente lo ejecutará una vez que el nodo base le dé la orden de hacerlo. El mecanismo de actualización puede fijar el momento en el que el nuevo *firmware* será ejecutado en los nodos de servicio y en ese momento, el nodo de servicio se reiniciará para comenzar a utilizar la nueva versión. Así, el nodo base podrá escoger entre reiniciar todos los nodos al mismo tiempo o hacerlo en varios pasos.

  Para indicar a los nodos de servicio el momento en el que deberán reiniciarse y comenzar a utilizar el nuevo *firmware*, el nodo base les enviará el paquete *FU_EXEC_REQ*. Este paquete incluye el campo *"Restar Timer"* que mediante su configuración el nodo base le indicará al nodo de servicio si debe reiniciarse de forma inmediata o el tiempo que debe esperar antes de hacerlo. El valor de este campo va desde 0 a 65536, por lo que el nodo de servicio esperará entre 0 y 65536 segundos antes de reiniciarse.

  Por otra parte, cuando un nodo de servicio indica al nodo base que ya ha recibido la imagen, el nodo base puede escoger entre solicitar al nodo de servicio que comience a utilizar el nuevo *firmware*, o puede esperar a lanzar dicha solicitud hasta que los demás nodos de la red también hayan recibido la imagen, y cuando esto ocurra, enviar las solicitudes a cada nodo de servicio de manera ordenada. El orden en el que se envían dichas solicitudes también es flexible.

- **Configuración del periodo de prueba del *firmware***. Después del reinicio, cada nodo de servicio ejecutará el nuevo *firmware* durante un periodo de prueba cuya duración puede ser previamente configurada por el nodo base. La duración de este periodo de prueba se indicará en el campo *"Safety Timer"* del paquete *FU_EXEC_REQ* y tendrá un valor entre 0 y 65536 segundos. Si el periodo de prueba llega a su fin y los nodos de servicio no reciben ninguna confirmación del nuevo *firmware* (paquete *FU_CONFIRM_REQ*) por parte del nodo base, o el nodo base decide abortar el proceso de actualización (mediante el envío del paquete *FU_KILL_REQ*), el nodo de servicio podrá descartar la nueva versión del *firmware* y volver a la versión antigua del mismo.

Teniendo en cuenta las posibilidades que el estándar deja abiertas a la hora de implementar la aplicación de actualización de *firmware*, en este trabajo de investigación se han diseñado, implementado y analizado diferentes estrategias de actualización de *firmware* con el fin de encontrar la mejor en términos de

disponibilidad de subred *PRIME*, sin que a cambio suponga un empeoramiento del tiempo de actualización de la red.

Para facilitar la descripción de las diferentes estrategias, cada estrategia se ha dividido en tres fases comunes que se describen a continuación. Es importante destacar que todas las estrategias implementadas funcionan en modo *multicast*, ya que en el modo de *unicast* la imagen de *firmware* se envía a cada uno de los nodos de servicio en la subred por separado y por lo tanto, el proceso se puede alargar en exceso. Por lo tanto, en las estrategias implementadas, todos los paquetes de control se envían a través de conexiones *unicast*, mientras que las páginas del *firmware* (datos) son enviadas mediante conexiones *multicast*.

La figura 5.2 muestra el diagrama de estados del mecanismo de actualización *firmware* así como los acontecimientos más relevantes en las transiciones de estados.

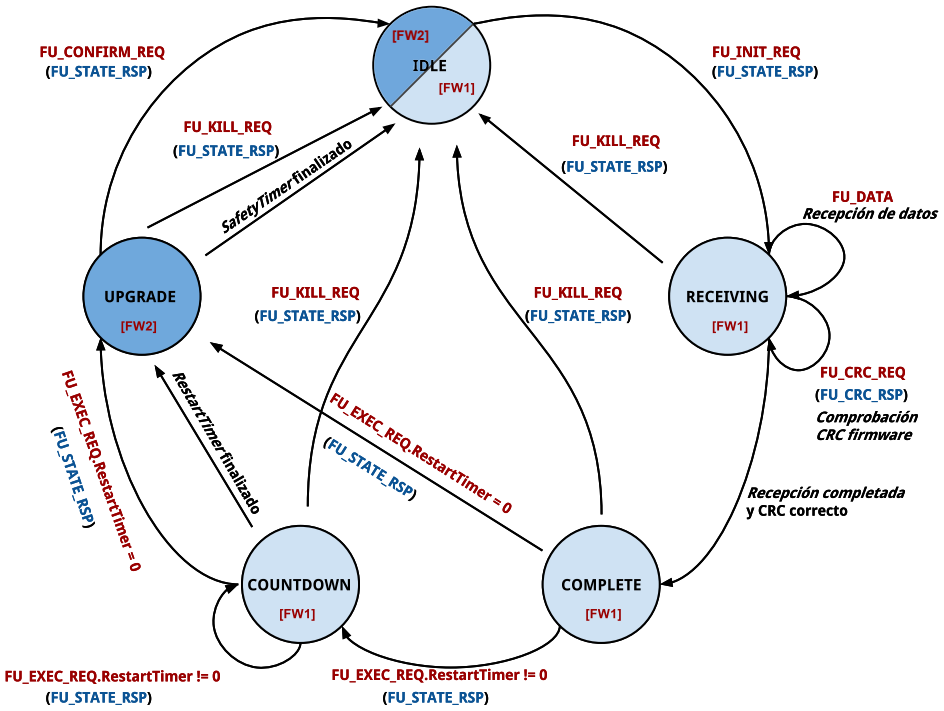


Figura 5.2: Diagrama de estados del proceso de actualización *firmware*

1. **Fase de inicialización**.

   La fase de inicialización del proceso de actualización se muestra en la figura 5.3 y consiste en tres acciones principales.

   - *Abrir conexión con un nodo de servicio*.

     El nodo base abre una conexión con un nodo de servicio mediante el envío del paquete de control *CON_REQ_B*, y para completar el

establecimiento de la conexión el nodo de servicio responderá con el paquete *CON_REQ_S*.

- *Enviar invitación al nodo de servicio para que se una al grupo multicast*.
  El nodo base invita al nodo de servicio a que se una al grupo *multicast* asignado. Para ello, le enviará el paquete de control *MUL_JOIN_B* que incluye el campo *MUL.LCID* donde se envía el identificador del grupo *multicast* al que deberá unirse. El nodo de servicio por su parte, en respuesta a la invitación de unirse al grupo *multicast* responderá con un paquete *MUL_JOIN_S*.

  Es importante mencionar que si el nodo de servicio de destino de este paquete se encuentra en un nivel mayor que 0 en la jerarquía, es decir, se ha registrado a través de otro nodo de servicio que actúa como *Switch*, cuando dicho *Switch* reciba el paquete de control *multicast*, añadirá una entrada en su tabla de *switching*. De este modo los paquetes que tengan una dirección de destino *multicast* y con un *LCID* igual al añadido en la tabla serán retransmitidos tal y como se ha explicado en la sección 2.2.3.3.4 del capítulo 2.

- *Enviar mensaje de inicialización del proceso de actualización firmware*.
  El nodo base le enviará un paquete de tipo *FU_INIT_REQ* para indicarle al nodo de servicio que la actualización está a punto de comenzar. Este paquete llevará información muy importante acerca del proceso como el tamaño total de la imagen del *firmware*, tamaño de las páginas del *firmware* y el CRC de la imagen, tal y como se ha explicado en la sección 2.2.4.2.4 del capítulo 2.

  Cuando el nodo de servicio recibe este paquete, siguiendo la máquina de estados de la figura 5.2, pasará del estado *"Idle"* a *"Receiving"* y quedará a la espera de recibir las páginas del *firmware*. Por otro lado, como respuesta a este paquete, el nodo de servicio le enviará un paquete *FU_STATE_RSP* al nodo base.

2. **Fase de descarga del *firmware***.

   Después de la fase de inicialización comienza la fase de descarga de la imagen del *firmware*. Existen distintas formas de llevar a cabo esta fase, y en las estrategias presentadas en este trabajo de investigación el nodo base comienza con la elección de un nodo de servicio aleatorio y la consulta de las páginas aún por recibir por dicho nodo. Después, en función de la respuesta recibida, se enviarán un número concreto de páginas y el nodo base volverá a escoger otro nodo y a realizar la misma consulta. Finalmente, cuando un nodo de servicio notifique al nodo base que ya ha recibido la imagen completa, éste le solicitará que compruebe la integridad del *firmware* y en caso de éxito le solicitará que abandone el grupo

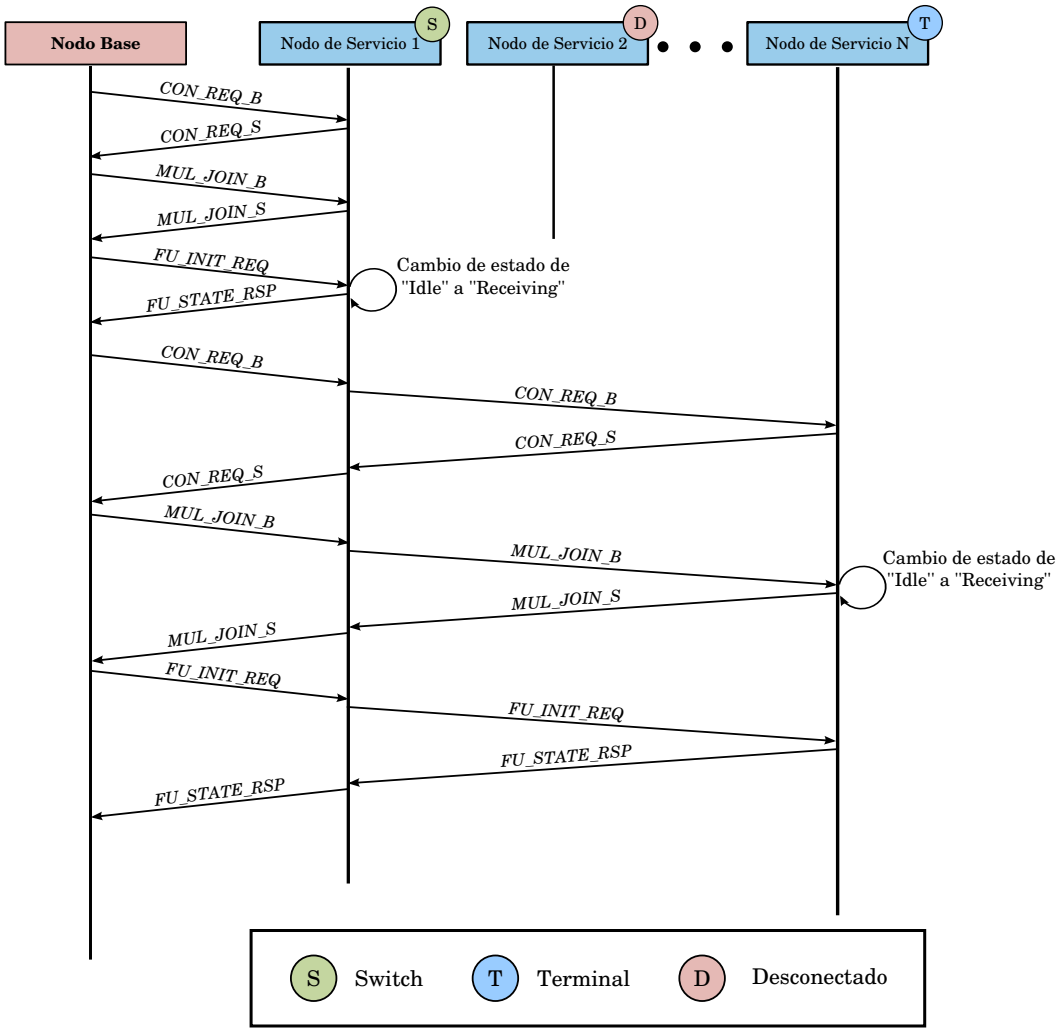


Figura 5.3: Diagrama de secuencia de la fase de inicialización

multicast. Por lo tanto, la fase de descarga del *firmware* que se muestra en la figura 5.4 se compone de las siguientes acciones.

- *Consultar páginas pendientes*.

  El nodo base escoge un nodo al azar para preguntarle por las páginas que le quedan aún por recibir. Para ello envía un paquete de tipo *FU_MISS_REQ* al nodo de servicio escogido y éste le responderá con un *FU_MISS_LIST* o *FU_MISS_BITMAP* en función del número de paquetes a enviar tal y como se explica en el apartado 2.2.4.2.4 del capítulo 2.

  Para la elección del nodo al azar, el nodo base escogerá un nodo de servicio de forma aleatoria entre aquellos nodos pendientes de actualizar y que ya hayan sido inicializados.

- *Enviar páginas del firmware*.

  En las estrategias implementadas, el nodo base comenzará con el envío de las páginas indicadas por el nodo de servicio, enviando hasta un máximo de 512 páginas con una separación temporal de 600 milisegundos. Tanto el número de páginas máximo a enviar como la separación temporal entre los envíos de páginas se configura antes del comienzo de la actualización *firmware*. Por otro lado, cada una de las páginas se enviará dentro de un paquete de tipo *FU_DATA* y el tamaño máximo de cada página también se definirá al comienzo del proceso de actualización. En este caso, para todas las estrategias implementadas en este trabajo de investigación, las páginas tendrán un tamaño de 64 bytes.

  Una vez que el nodo base haya enviado las páginas solicitadas por el primer nodo de servicio consultado, el nodo base volverá a escoger un nuevo nodo de servicio al que preguntar por sus páginas pendientes.

  Como ya se ha comentado, el estándar PRIME indica (en sus ejemplos) que justo después de enviar el paquete de inicialización y haber recibido la respuesta correspondiente por parte del nodo de servicio, el nodo base comienza a enviar las páginas del *firmware*. Sin embargo, como los nodos de servicio puede que hayan recibido algunas páginas del *firmware* en un proceso de actualización anterior sin finalizar, en las estrategias implementadas la fase de descarga del *firmware* comienza con la elección de un nodo de servicio al azar al que preguntarle por la lista de páginas que le quedan por recibir. En función de la respuesta de dicho nodo, se enviarán las páginas del *firmware* que irán dirigidas al grupo *multicast* al que pertenece el nodo consultado.

  La razón por la que se escoge un nodo de servicio al azar para realizar dicha consulta es que, teniendo en cuenta que las páginas del *firmware* van dirigidas al grupo *multicast* en lugar de a un solo nodo de servicio, se asume que las páginas que le faltan al nodo consultado serán las mismas que echaran en falta los demás nodos integrantes del mismo grupo *multicast*. De este modo se logra reducir el número de consultas a los nodos, disminuyendo así el tráfico de paquetes de control para evitar la saturación de la red.

- *Solicitar comprobación de integridad del firmware*.

  Cuando un nodo de servicio haya recibido la imagen del *firmware* por completo, el nodo base le enviará un paquete *FU_CRC_REQ* para que el nodo de servicio compruebe la integridad del mismo. Si el *firmware* se ha recibido correctamente, el nodo contestará con el

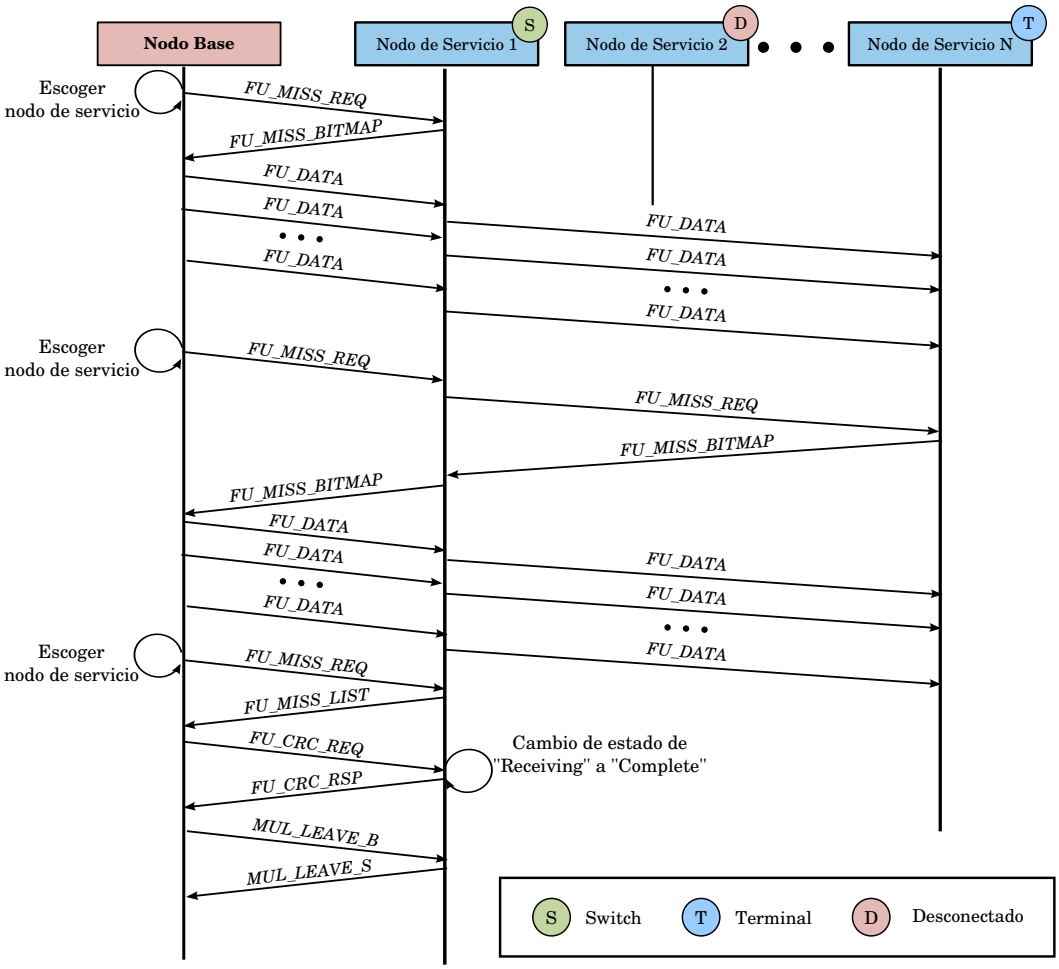


Figura 5.4: Diagrama de secuencia de la fase de descarga del *firmware*

paquete de respuesta correspondiente (*FU_CRC_RSP*) y pasará al estado *"Complete"*. De lo contrario, el nodo de servicio notificará del error al nodo base mediante el paquete *FU_CRC_RSP*, descartará la imagen recibida, actualizará el *bitmap* donde guarda la información acerca de las páginas pendientes y quedará a la espera de volver a recibir las páginas del *firmware*.

- *Invitar al nodo de servicio a que abandone el grupo multicast*.

  Después de la correcta recepción de la imagen del *firmware*, el nodo base invitará al nodo de servicio a que abandone el grupo *multicast* enviándole un paquete de tipo *MUL_LEAVE_B*. El nodo de servicio le responderá mediante un paquete *MUL_LEAVE_S*.

3. **Fase de activación**.

   La fase de activación del *firmware* se muestra en la figura 5.5 y consiste en tres acciones principales que se detallan a continuación.

   - *Ejecutar el nuevo firmware*.

     Al comienzo de la fase de activación, el nodo base envía un paquete de tipo *FU_EXEC_REQ* al nodo de servicio para indicarle el momento en el que tendrá que reiniciarse y comenzar a utilizar el nuevo *firmware*. Como respuesta a este paquete, el nodo de servicio contestará con un *FU_STATE_RSP*.

     El instante de reinicio vendrá indicado por el campo *"Restart Timer"* del paquete *FU_EXEC_REQ*. En caso de que el valor de dicho campo sea 0, tal y como ocurre en las estrategias implementadas, el nodo de servicio se reiniciará inmediatamente, pasará al estado *"Upgrade"* y comenzará a utilizar el nuevo *firmware*. En cambio, si el valor de dicho campo es mayor que 0 el nodo de servicio pasará al estado *"Countdown"* y esperará el tiempo indicado por el campo *"Restart Timer"* antes de reiniciarse, pasar al estado *"Upgrade"* y comenzar a utilizar el nuevo *firmware*.

   - *Enviar confirmación de nuevo firmware*.

     El nuevo *firmware* se utilizará durante el periodo de prueba establecido por el campo *"Safety Timer"* del paquete *FU_EXEC_REQ* y el nodo quedará a la espera de recibir la confirmación del nuevo *firmware* por parte del nodo base que llegará con el paquete *FU_CONFIRM_REQ*. Cuando llegue la confirmación, como respuesta a este paquete el nodo de servicio enviará un paquete del tipo *FU_STATE_RSP*, cambiará su estado a *"Idle"* y descartará la versión antigua del *firmware*. En cambio, si el nodo de servicio no recibe ningún paquete de confirmación antes que transcurra el tiempo indicado por el campo *"Safety Timer"*, el nodo de servicio descartará el nuevo *firmware*, se reiniciará y volverá a la versión anterior.

   - *Cerrar conexión con nodo de servicio*.

     Una vez que el nodo base haya terminado con la activación del *firmware* de un nodo de servicio, cerrará la conexión establecida con él mediante el envío del paquete *CON_CLOSE_B*. El nodo de servicio contestará con un *CON_CLOSE_S*.

Finalmente, como se puede observar en el diagrama de estados de la figura 5.2, el nodo base puede terminar el proceso de actualización del *firmware* en cualquier momento mediante el envío de un paquete de tipo *FU_KILL_REQ*. Cuando un nodo de servicio recibe este paquete, el nodo de servicio cambiará al

estado *"Idle"* y opcionalmente borrará los datos descargados. El nodo de servicio responderá con un paquete *FU_STATE_RSP*.

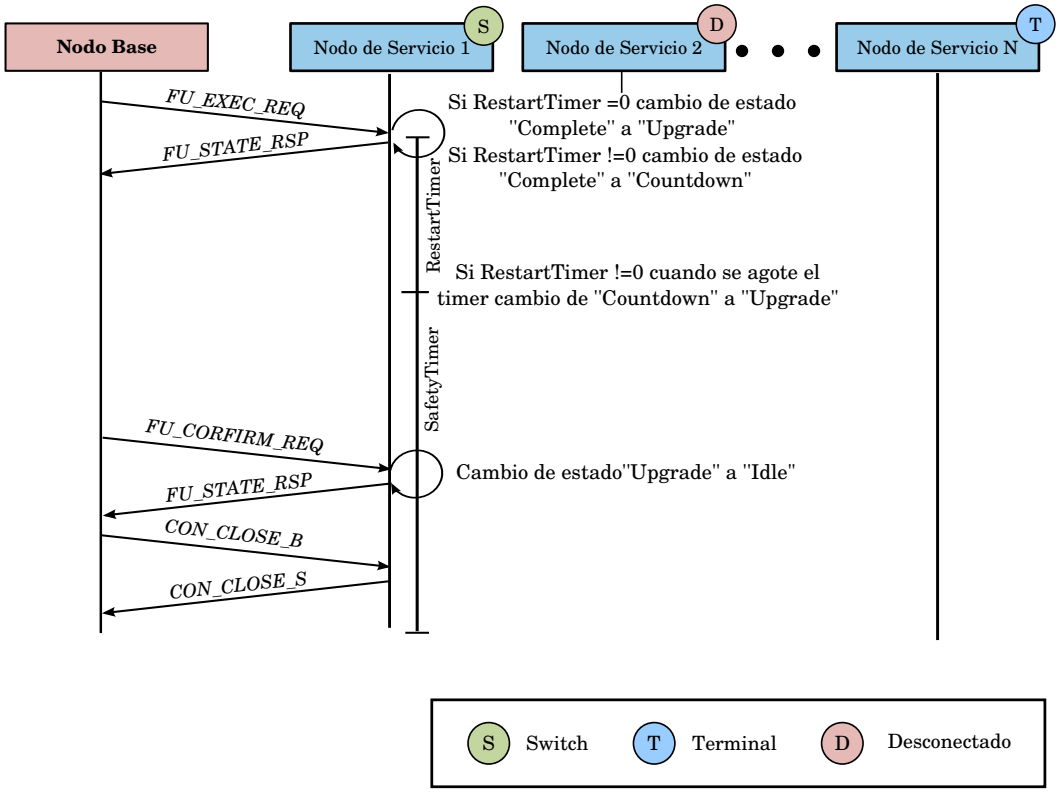


Figura 5.5: Diagrama de secuencia de la fase de activación del *firmware*

5.3 Estrategias de actualización *firmware*

En esta sección, se describen los criterios seguidos a la hora de diseñar las estrategias de actualización, así como el funcionamiento de las mismas en detalle.

5.3.1 Criterios de diseño de las estrategias de actualización

En la validación del modelo de simulación de subredes PRIME descrito en el capítulo 4 [LnLUS15] se empleó la estrategia de actualización mostrada en la figura 5.7 la cual fue implementada tanto en el entorno real como en el de simulación. A esta estrategia la denominaremos estrategia A y se explicará en detalle en el siguiente subapartado.

Tabla 5.1: Resumen de los resultados de la validación del modelo de simulación de subredes PRIME

Parámetro	**Escenario 1**	**Escenario 2**	**Escenario 3**
Núm. nodos de servicio	68	45	69
Núm. nodos de servicio a actualizar	58	33	58
Núm. total de niveles en la jerarquía	4	4	2
Profundidad	3	3	1
Anchura	3	6	3
Tiempo de actualización (simulación)	2 h 42 min	3 h 22 min	1 h 23 min
Tiempo de actualización (campo)	2 h 54 min	3 h 23 min	1 h 19 min

Los resultados de dicho trabajo muestran que la duración del proceso de actualización *firmware* de la subred no está directamente relacionada con el número de nodos de servicio a actualizar.

La tabla 5.1 muestra un resumen de los resultados de los experimentos que se llevaron a cabo en dicho trabajo. Los escenarios se describen en función de los siguientes parámetros: número total de nodos de servicio en la subred, número de nodos de servicio a actualizar, número de niveles en la jerarquía y la profundidad y anchura de la subred. Hay que tener en cuenta que, la profundidad de la red se define como el número total de niveles que hay en la jerarquía menos uno, mientras que la anchura se define como el máximo número de *Switches* que puede haber en el nivel 0 de la subred (número de ramas). La figura 5.6 muestra un ejemplo de una topología lógica de una subred en la que se indican su anchura y profundidad.

Como se puede observar en la tabla, a medida que aumenta la profundidad y anchura de la subred los resultados de tiempo de actualización se vuelven significativamente peores. Por ejemplo, en los escenarios 1 y 3 hay 58 nodos de servicio a actualizar, pero en el escenario 1, donde la profundidad de la red es de 3, el tiempo de actualización es bastante mayor que en el escenario 3, donde la profundidad de la red es de 1. Por otro lado, en el escenario 2, donde solamente hay 33 nodos de servicio a actualizar, la profundidad es la misma que en el escenario 1 y la anchura de la subred es mayor, el proceso de actualización tarda más. Por lo tanto, se puede concluir que la topología lógica de la red tiene una influencia significativa en el comportamiento del proceso de actualización *firmware*.

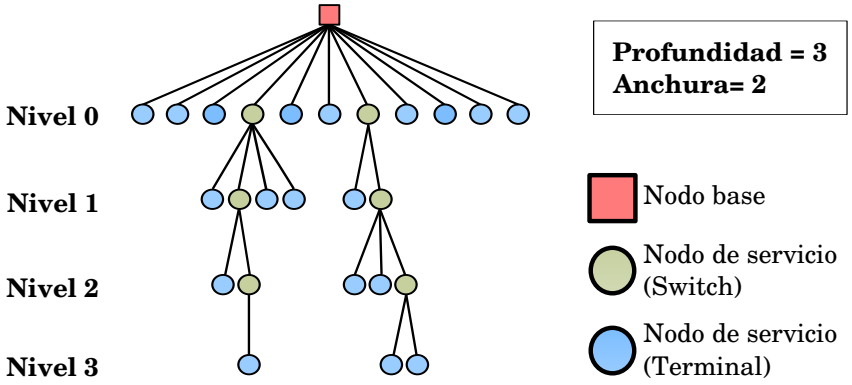


Figura 5.6: Ejemplo de topología lógica incluyendo la información de anchura y profundidad

La estrategia A que ha sido empleada en los experimentos de la validación del modelo de simulación, se caracteriza por el hecho de que cuando un nodo de servicio notifica al nodo base que ya ha recibido las páginas del *firmware* correctamente, el proceso pasa directamente a la fase de activación del *firmware* de dicho nodo. Por lo tanto, el orden de activación del *firmware* de los nodos de servicio es aleatorio, es decir, la estrategia A no toma en cuenta ningún aspecto de la topología lógica de la red a actualizar. Esto significa que el reinicio de los nodos es también aleatorio, por lo que la disponibilidad de la red puede verse afectada negativamente.

Teniendo en cuenta la influencia que tiene la topología lógica sobre el comportamiento del proceso de actualización y aprovechando la flexibilidad del estándar PRIME, se han diseñado, implementado y analizado cuatro estrategias de actualización más. El diseño de estas estrategias se ha realizado tomando en cuenta aspectos de la topología lógica de las subredes y serán explicadas en detalle en el siguiente apartado junto con la estrategia A. En concreto, para el diseño de las estrategias se ha puesto especial atención en la activación del *firmware*, ya que se considera que es la fase que mayor influencia puede tener sobre los resultados de disponibilidad de la subred.

Después de haber implementado las estrategias, éstas han sido analizadas y comparadas en redes con diferentes topologías físicas y lógicas, con el fin de encontrar la mejor estrategia en términos disponibilidad de red, sin que por otra parte suponga un aumento de la duración del proceso de actualización. Por lo tanto, en la evaluación de estas estrategias se han empleado dos métricas: la disponibilidad PRIME de la subred y el tiempo de actualización *firmware*. Como ya se ha descrito al comienzo de este capítulo, la disponibilidad PRIME de un nodo de servicio [SBA+13] se define como la cantidad de tiempo durante el cual un nodo de servicio se encuentra en el estado *Terminal* o *Switch* (estado

disponible) comparándolo con el tiempo total de observación, mientras que la disponibilidad de la subred es el valor medio de las disponibilidades de los elementos de la subred. El tiempo de observación por su parte, es la duración del proceso de actualización *firmware*.

5.3.2 Estrategias de actualización *firmware* implementadas

A continuación, se explica el funcionamiento de las estrategias de actualización *firmware* implementadas.

5.3.2.1 Estrategia A

Al comienzo del proceso de actualización, el nodo base define un único grupo *multicast* en el que se incluirá a todos los nodos de servicio del mismo fabricante y que ejecutan el mismo *firmware*. Después, el nodo base escoge de forma aleatoria un nodo que esté en un estado disponible (*Terminal* o *Switch*) y procede con su inicialización. Este proceso se repite con todos los nodos de servicio que formarán parte del grupo *multicast*. Los nodos que no hayan podido ser inicializados esperarán a la siguiente ronda o intento de actualización para ser actualizados.

Después de haber completado la fase de inicialización, comienza la fase de descarga del *firmware* en la que el nodo base escoge un nodo al azar entre los nodos inicializados y le pregunta por las páginas del *firmware* que le quedan por recibir para completar la imagen. Al recibir la respuesta, el nodo base comienza con el envío de las páginas solicitadas por el nodo de servicio. El destino de dichas páginas será el grupo *multicast* al que pertenece el nodo de servicio consultado. Una vez que el nodo base haya enviado las páginas solicitadas (hasta 512 páginas), el nodo base escoge otro nodo de servicio al azar y le pregunta por las páginas que le quedan por recibir.

Cuando un nodo de servicio notifica al nodo base que ya ha recibido todas las páginas, el nodo base le solicita que compruebe la integridad del *firmware*. En caso de que la imagen sea correcta, el nodo base le indica al nodo de servicio que abandone el grupo *multicast* y comienza con la fase de activación del *firmware* de dicho nodo. Cuando se completa esta fase, el proceso vuelve a la fase de descarga del *firmware* en la que el nodo base nuevamente escoge un nodo de servicio que aún no ha sido actualizado (pero está inicializado) al azar y le pregunta por las páginas que le quedan por recibir. Si no quedan nodos inicializados a los que consultar por sus páginas, el proceso de actualización volverá al punto inicial donde el nodo base tratará de inicializar los nodos de servicio que aún no hayan sido actualizados. La figura 5.7 muestra el funcionamiento de la estrategia A.

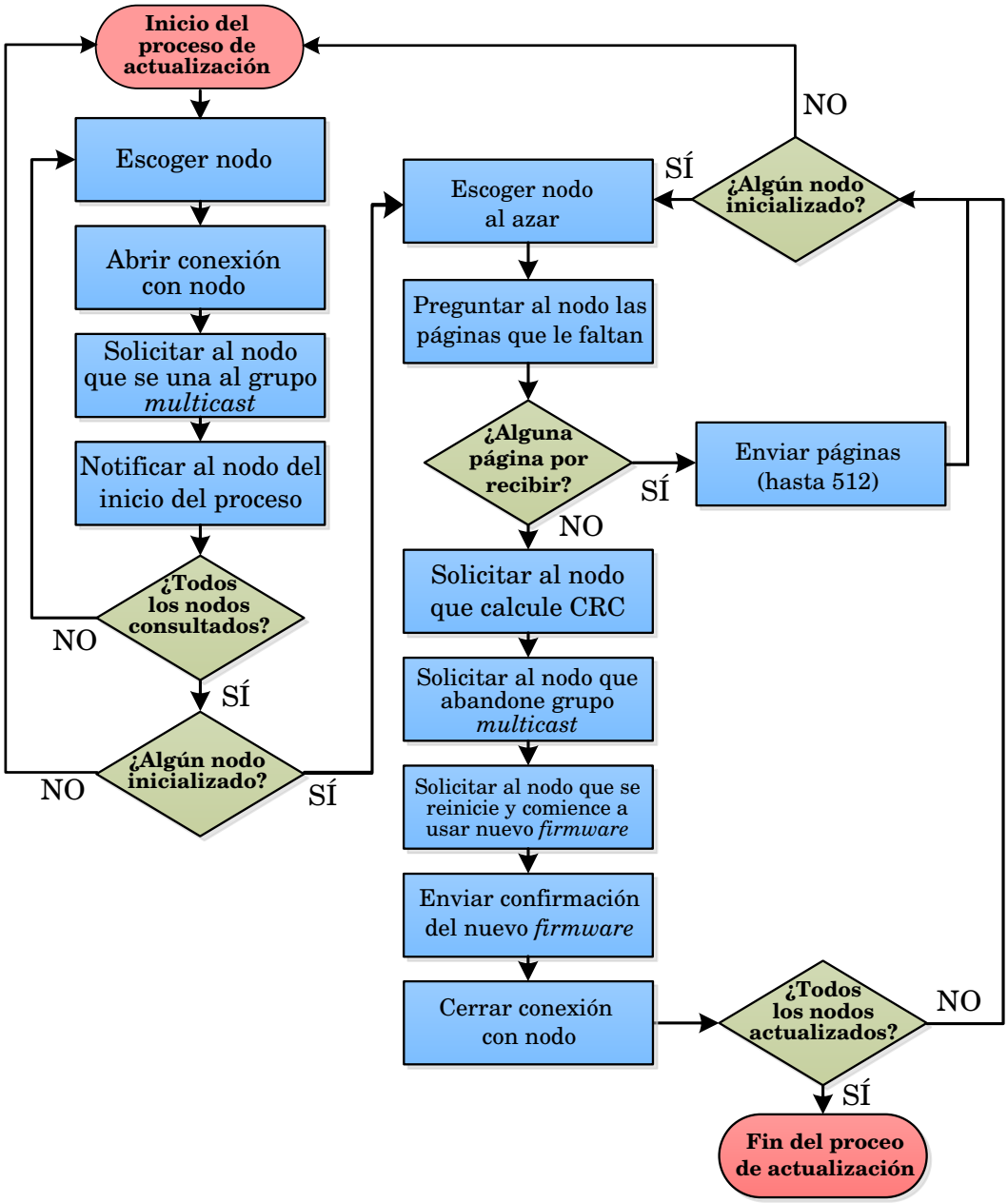


Figura 5.7: Diagrama de flujo que representa el funcionamiento de la estrategia de actualización A (empleada en el proceso de validación del modelo de simulación)

5.3.2.2 Estrategia B

En esta estrategia, al comienzo del proceso de actualización, el nodo base define distintos grupos *multicast* en función del nivel lógico de jerarquía de los nodos de servicio.

Una vez que los grupos *multicast* ya han sido definidos, el nodo base comienza con la actualización *firmware* de los distintos grupos *multicast* en orden descendente, es decir, comenzando por los nodos de servicio más alejados del nodo base hasta llegar a los más cercanos. Para realizar la actualización de cada uno de los grupos *multicast*, se ha seguido la estrategia A donde el *firmware* de cada uno de los nodos de servicio se activa inmediatamente después de que los nodos de servicio notifican al nodo base que han recibido la imagen del *firmware* correctamente.

Cuando el nodo base termina con la actualización de los nodos de servicio del primer grupo *multicast*, tratará de actualizar los nodos que componen el siguiente grupo *multicast* que cuentan un nivel de jerarquía menor (más cercanos al nodo base). Sin embargo, si el nodo base no puede actualizar uno o más nodos del grupo *multicast* después de varios intentos, se pospone su actualización para la siguiente ronda y se continúa con la actualización de los nodos de servicio del siguiente grupo.

En esta estrategia, la activación del *firmware* es inmediata, así que tan pronto como un nodo de servicio recibe todas páginas del *firmware* correctamente, el nodo base le pide al nodo que comience a usar el nuevo *firmware*. La figura 5.8 muestra el diagrama de flujo de la estrategia de actualización B.

5.3.2.3 Estrategia C

En esta estrategia, el nodo base define un único grupo *multicast* en el que se incluyen todos los nodos de servicio del mismo fabricante, tal y como ocurre en la estrategia A.

El proceso comienza con la fase de inicialización, durante la cual el nodo base trata de inicializar el mayor número de nodos de servicio. La actualización de los nodos que no han podido ser inicializados se deja para la siguiente ronda de actualización.

Al terminar la fase de inicialización, comienza la fase de descarga del *firmware*. En esta estrategia, cuando un nodo de servicio indica al nodo base que ha recibido todas las páginas del *firmware* correctamente, en lugar de proceder con la activación del mismo, el nodo base almacena el identificador del nodo de servicio en un vector. Después, cuando todos los nodos de servicio inicializados le notifican al nodo base que han recibido la imagen correctamente, el nodo base ordena los identificadores de los nodos de servicio en función de su nivel en la jerarquía en orden descendente y comienza con la activación del *firmware* en dicho orden.

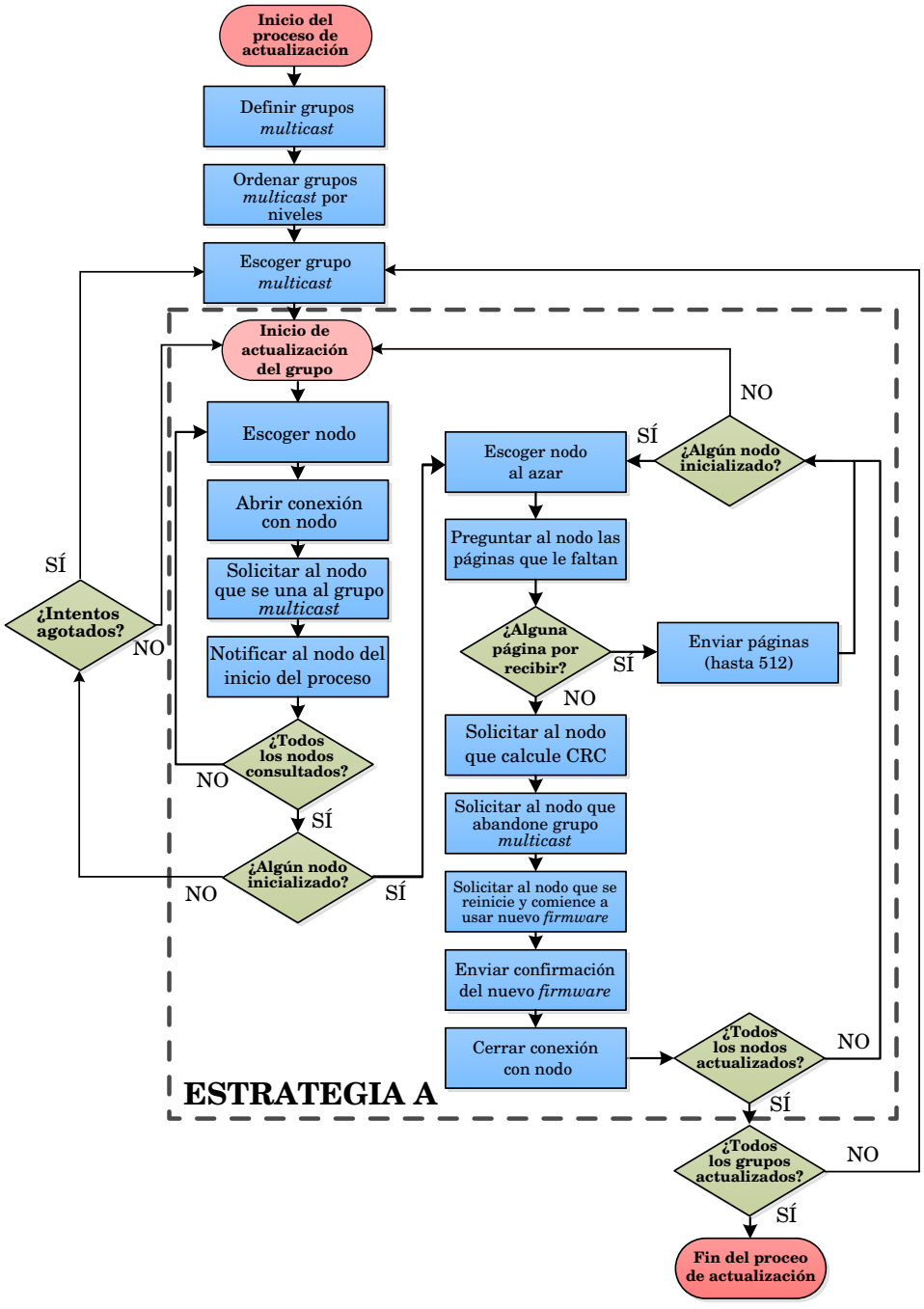


Figura 5.8: Diagrama de flujo que representa el funcionamiento de la estrategia de actualización B

Una vez que el *firmware* de los nodos ha sido activado, el nodo base comprueba si aún quedan nodos por actualizar. Si es así, el proceso de *firmware* vuelve al punto de partida y el nodo base trata de actualizar los nodos restantes. En cambio, si ya no quedan nodos por actualizar, el proceso de actualización *firmware* llega a su fin. La figura 5.9 muestra el diagrama de flujo de la estrategia de actualización C.

5.3.2.4 Estrategia D

En esta estrategia, el nodo de base define un único grupo de *multicast* en el que se incluirán todos los nodos de servicio del mismo fabricante.

Cuando un nodo de servicio notifica al nodo base que ya ha recibido todas las páginas de la imagen del *firmware* correctamente, el nodo base comprueba si el nodo de servicio se encuentra en el estado *Terminal* o *Switch*. En caso de que se trate de un nodo *Terminal*, la imagen del *firmware* será activada inmediatamente, pero si es un nodo *Switch*, el nodo base esperará hasta que todos los nodos *Terminales* inicializados en la actual ronda de actualización de *firmware* hayan sido actualizados. Así, tan pronto como todos los nodos *Terminales* inicializados hayan sido activados, el nodo base activará el *firmware* de los nodos *Switch*. Después, el nodo base comprobará si todos los nodos de servicio del grupo *multicast* ya han sido actualizados y en caso de no ser así, el proceso volverá al punto de partida de modo que el nodo base tratará de actualizar los nodos de servicio restantes. La figura 5.10 muestra el diagrama de flujo de la estrategia D.

5.3.2.5 Estrategia E

En esta estrategia al comienzo del proceso de actualización el nodo base define un único grupo *multicast* en el que se incluyen todos los nodos de servicio del mismo fabricante, y comienza con la fase de inicialización de los nodos que formarán parte de dicho grupo. Así, cuando los nodos ya han sido inicializados el nodo base comenzará con la fase de descarga de *firmware*.

Cuando un nodo de servicio notifica al nodo base que ya ha recibido todas las páginas del *firmware* correctamente, el nodo base comprueba si el nodo de servicio es un *Terminal* o un *Switch*. Si se trata de un *Terminal* el nodo base comenzará con la fase de activación del *firmware* de dicho nodo. Sin embargo, en caso de que sea un *Switch* el nodo base comprobará si los nodos del grupo *multicast* que cuelgan directamente de dicho nodo (hijos) ya han sido actualizados antes de proceder con la activación del *firmware* del *Switch*. Si los hijos del *Switch* no han sido actualizados aún, el nodo base volverá a la fase de descarga del *firmware* y continuará enviando las páginas que aún les faltan por recibir al resto de los nodos de servicio. Por lo tanto, en esta estrategia, el nodo base no

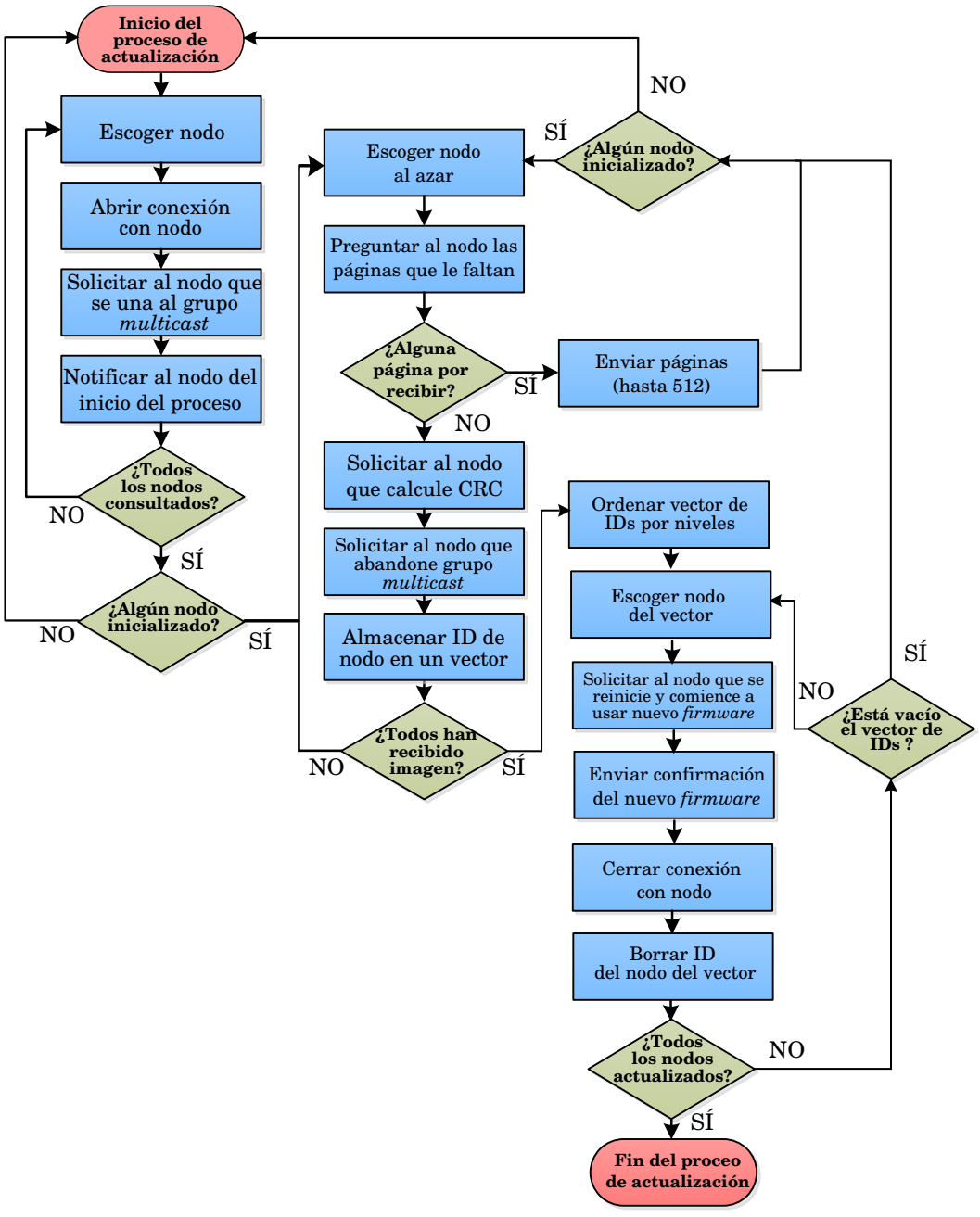


Figura 5.9: Diagrama de flujo que representa el funcionamiento de la estrategia de actualización C

activará el *firmware* de los nodos *Switch* hasta que sus hijos hayan sido actualizados. La figura 5.10 muestra el funcionamiento de la estrategia E descrita.

5.4 Evaluación de las estrategias

Una vez que se ha realizado el diseño de las distintas estrategias de actualización, se han implementado y evaluado mediante el modelo de simulación de subredes *PRIME* desarrollado en este trabajo de investigación (ver capítulo 4) [LnLUS15].

Para llevar a cabo la evaluación, en primer lugar, se han definido los distintos escenarios sobre los que se van a ejecutar dichas estrategias de actualización. Se han definido tres redes de distinto tamaño y topología física conocidas como *Rural*, *Residencial tipo 1* y *Residencial tipo 2* tal y como se explicará a continuación. Estos escenarios son ejemplos de redes de baja tensión europeas.

- ***Rural***. Se trata de una red compuesta por 10 nodos de servicio dispuestos a lo largo de una línea. Cada uno de los nodos de servicio representa un punto de suministro o contador situado en una vivienda familiar como suele ser habitual en los entornos rurales. Los nodos de servicio están separados por unos cientos de metros el uno del otro siguiendo un esquema irregular. La figura 5.11 muestra la apariencia de una red de este tipo.

- ***Residencial tipo 1***. Se trata de una red compuesta por 32 nodos de servicio. También en este caso, cada uno de los nodos representa un punto de suministro o contador situado en una vivienda familiar de una urbanización. La red está compuesta por 4 líneas de alimentación o *Low Voltage (LV) feeders* conectados a un mismo transformador, mientras que en cada línea hay 8 nodos de servicio. La figura 5.12 muestra la apariencia de una red de este tipo.

- ***Residencial tipo 2***: Se trata de una red compuesta por 66 nodos de servicio. Cada uno de los nodos representa un punto de suministro o contador. La red se compone de 6 LV *feeders* y a cada una de estas líneas se conectan 11 nodos de servicio. La figura 5.13 muestra la apariencia de una red de este tipo.

Para cada uno de estos escenarios, se han definido distintas topologías lógicas con diferente anchura y profundidad. Tal y como se ha mencionado en el apartado 5.3.1 de este capítulo, la profundidad de una red es el número de niveles que hay en la jerarquía menos uno, mientras que la anchura se define como el número máximo de *Switches* que puede haber en el nivel 0 de la subred. La tabla 5.2 recoge las distintas topologías lógicas definidas para cada escenario descrito.

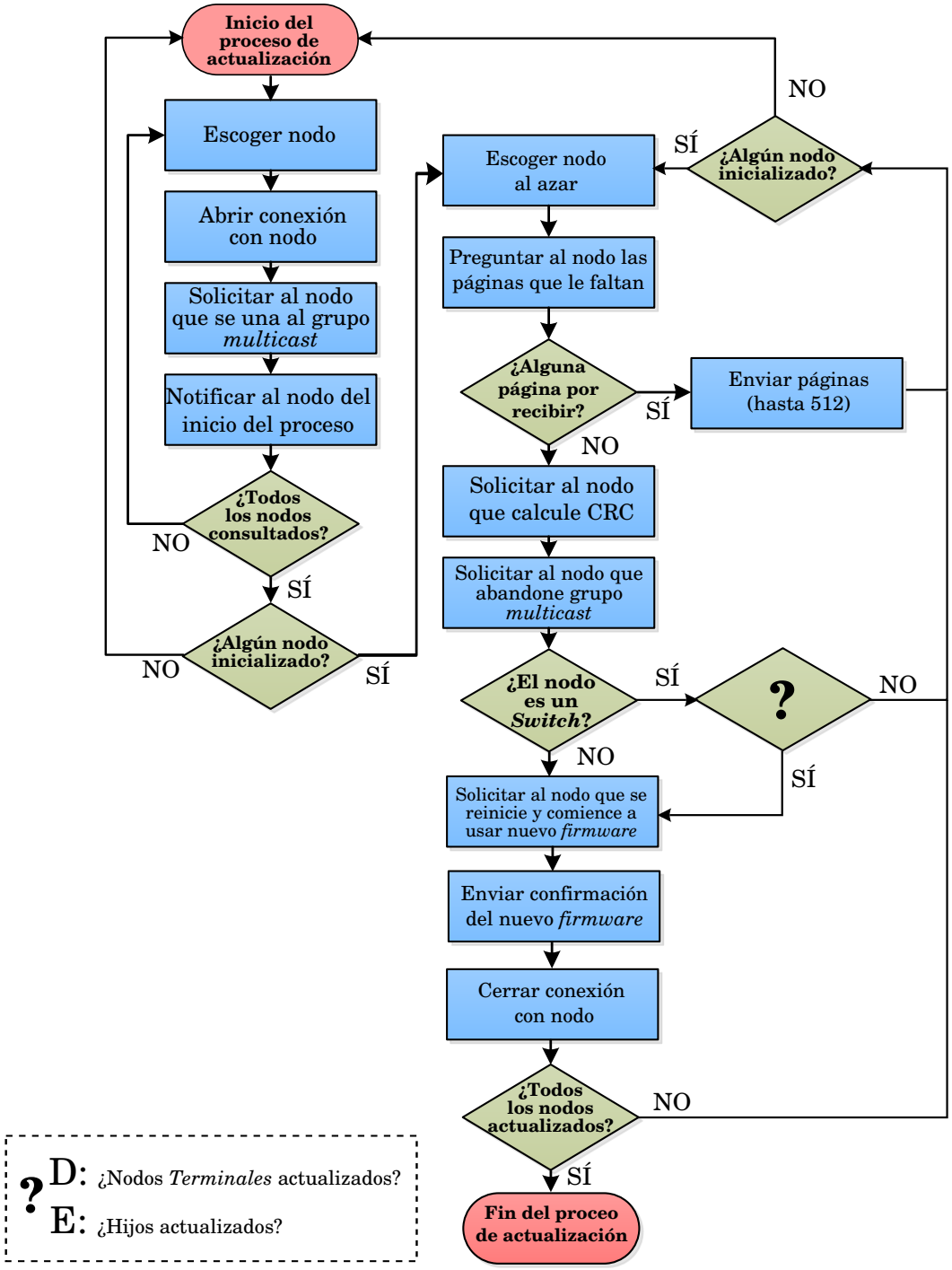


Figura 5.10: Diagrama de flujo que representa el funcionamiento de las estrategias de actualización D y E

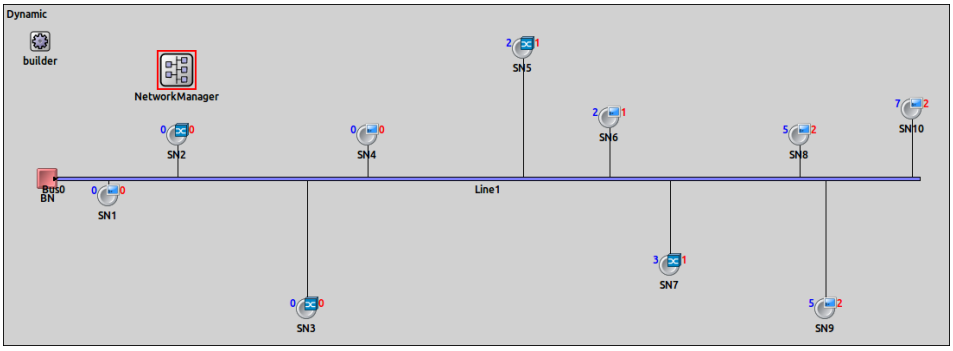


Figura 5.11: Escenario *Rural* en el entorno de simulación

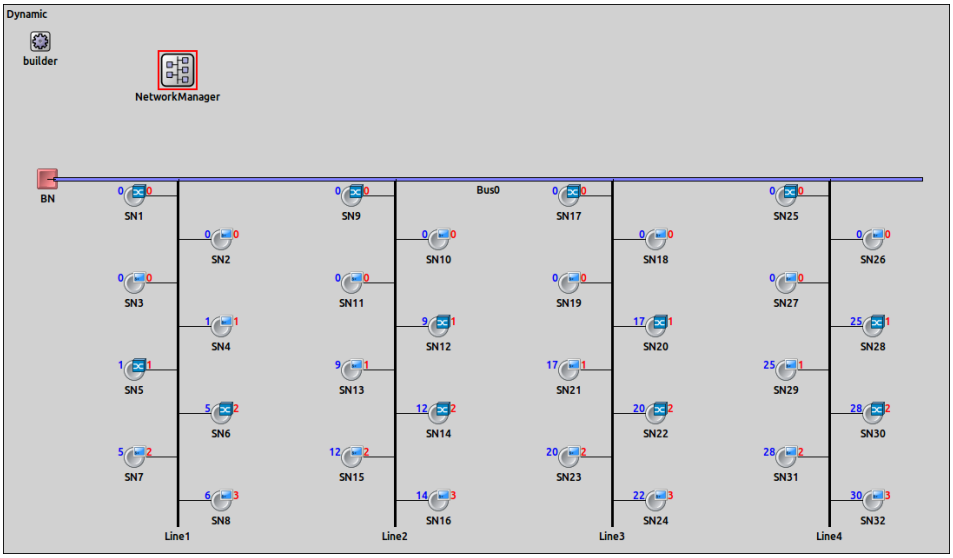


Figura 5.12: Escenario *Residencial tipo 1* en el entorno de simulación

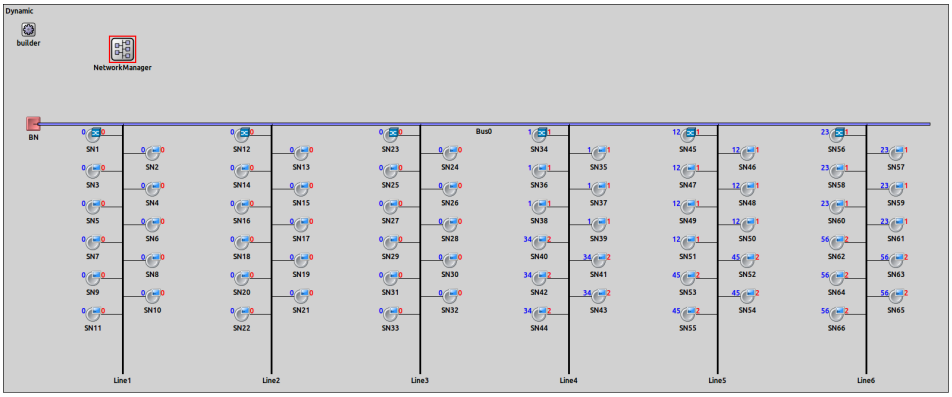


Figura 5.13: Escenario *Residencial tipo 2* en el entorno de simulación

Tabla 5.2: Topologías lógicas definidas para cada escenario

		Rural				Residencial tipo 1				Residencial tipo 2			
		Anchura				Anchura				Anchura			
		0	1	2	3	0	1	2	4	0	1	2	3
Profundidad	0	✓				✓				✓			
	1		✓	✓	✓		✓	✓	✓		✓	✓	✓
	2		✓	✓	✓		✓	✓	✓		✓	✓	✓
	3		✓	✓	✓		✓	✓	✓		✓	✓	✓
	4						✓	✓	✓		✓	✓	✓

En la siguiente sección, se muestran los resultados obtenidos de las simulaciones en cada uno de los escenarios descritos.

5.4.1 Resultados de las simulaciones

El objetivo de este trabajo de investigación es mejorar la disponibilidad de subredes PRIME durante el proceso de actualización *firmware* sin aumentar la duración del proceso. Para ello, se han diseñado e implementado distintas estrategias de actualización *firmware* que toman en cuenta la topología lógica de la red. Por lo tanto, en el análisis de estas estrategias se han utilizado dos métricas: la disponibilidad de la subred y la duración del proceso de actualización.

Es importante mencionar que el proceso de actualización *firmware* termina justo después de que el nodo base termine con la activación del *firmware* del último nodo de servicio a actualizar. Por otra parte, la disponibilidad PRIME de un nodo de servicio, tal y como se define en [SBA+13], hace referencia a la cantidad de tiempo durante el cual un nodo de servicio se encuentra en el estado *Terminal* o *Switch* (estado disponible) comparándolo con el tiempo de observación. La disponibilidad *PRIME* de la subred es por tanto el valor medio de las disponibilidades de los nodos de servicio de la red. La disponibilidad de un nodo de servicio es un valor porcentual y se calcula tal y como muestra la ecuación 5.1.

$$\text{Disponibilidad de nodo} = \frac{\text{Tiempo en estado disponible}}{\text{Tiempo de observación}} * 100\ (\%) \qquad (5.1)$$

Los resultados de los experimentos se han agrupado por tipos de escenario (*Rural*, *Residencial tipo 1* y *Residencial tipo 2*) y dentro de cada escenario se han realizado experimentos sobre distintas topologías lógicas con distinta anchura y profundidad (ver tabla 5.2). Además, con el fin de obtener resultados estadísticamente significativos, cada tipo de experimento, es decir, cada estrategia en

Tabla 5.3: Parámetros de simulación

Parámetro	Valor
Tamaño del *firmware*	98.432 ≈ 100 kBytes
Tamaño de página	64 Bytes
Número de páginas	1538 páginas
Máx. páginas a enviar	512 páginas
Espaciado entre páginas	600 milisegundos
Restart Timer	0 segundos
Safety Timer	32400 segundos
Esquema de modulación	DBPSK con FEC

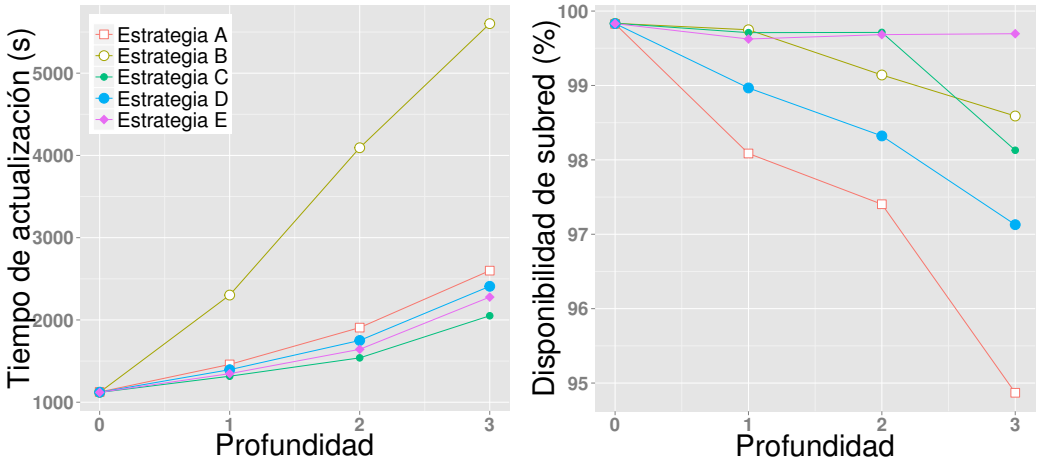
cada tipo escenario con una anchura y profundidad dada, se ha ejecutado 30 veces. Así, cada uno de los puntos mostrado en los gráficos representa el valor medio de 30 repeticiones del experimento indicado. La tabla 5.3 muestra los valores de algunos de los parámetros de la aplicación de actualización *firmware* más importantes que se han empleado en las simulaciones.

Los resultados del escenario *Rural* se muestran en las figuras 5.14, 5.15 y 5.16. Los resultados del escenario *Residencial tipo 1* se muestran en las figuras 5.17, 5.18 y 5.19. Finalmente, los resultados para el escenario *Residencial de tipo 2* se muestran en las figuras 5.20, 5.21 y 5.22. Las subfiguras (a) muestran el tiempo de actualización de los nodos de la subred mientras que las subfiguras (b) muestran la disponibilidad de la subred, para cada una de las estrategias de actualización, escenario y topología lógica. Estos gráficos muestran la tendencia de los resultados de las estrategias a medida que la anchura y la profundidad de las subredes aumenta.

Con el objetivo de poder analizar y comparar los resultados en profundidad, especialmente en los casos en los que las diferencias entre las estrategias son menos evidentes, los resultados mostrados en los gráficos se han transferido a las tablas 5.4 y 5.5.

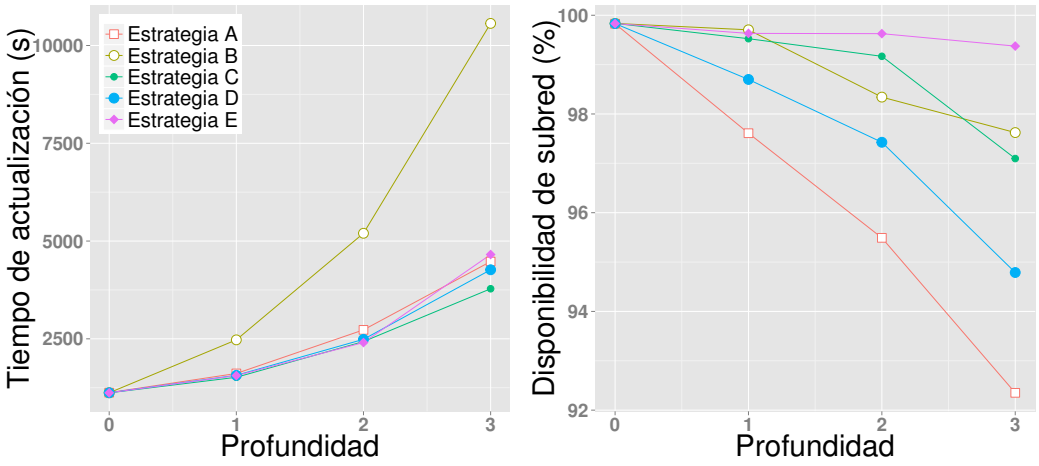
La tabla 5.4 muestra la media de los valores de disponibilidad en el eje de profundidad para cada gráfico, de manera que los resultados de las diferentes estrategias se pueden comparar entre sí de una manera más fácil. El cálculo de esta media nos da una visión general de los resultados de las diferentes estrategias y nos ayuda a evaluar los resultados que son más difíciles de distinguir a simple vista en los gráficos. Por otro lado, se ha realizado la misma tarea con los datos de tiempo de actualización y las medias calculadas (también en el eje de profundidad) se muestran en la tabla 5.5.

Después de calcular los valores medios, en función del resultado, cada estrategia se ha puntuado usando los siguientes criterios. En el caso de los resultados



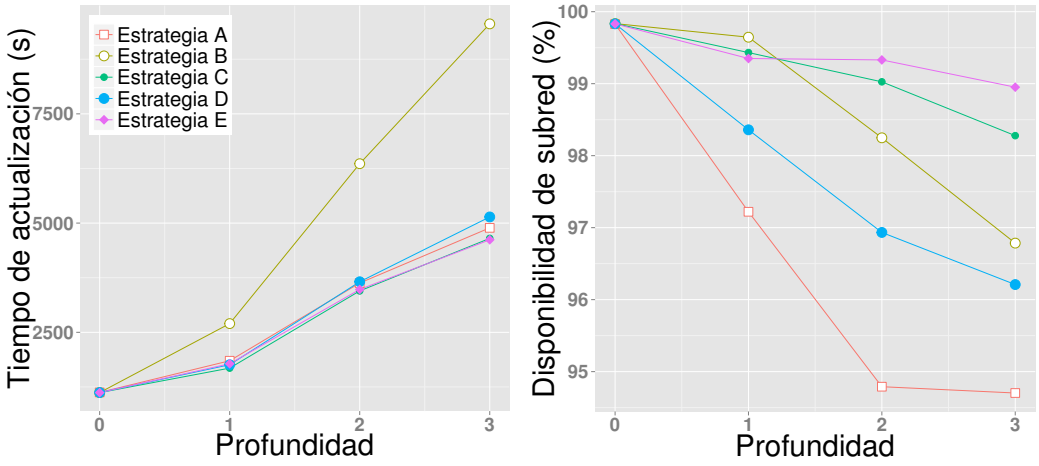
(a) Tiempo de actualización (b) Disponibilidad

Figura 5.14: Resultados en el escenario *Rural* de anchura 1



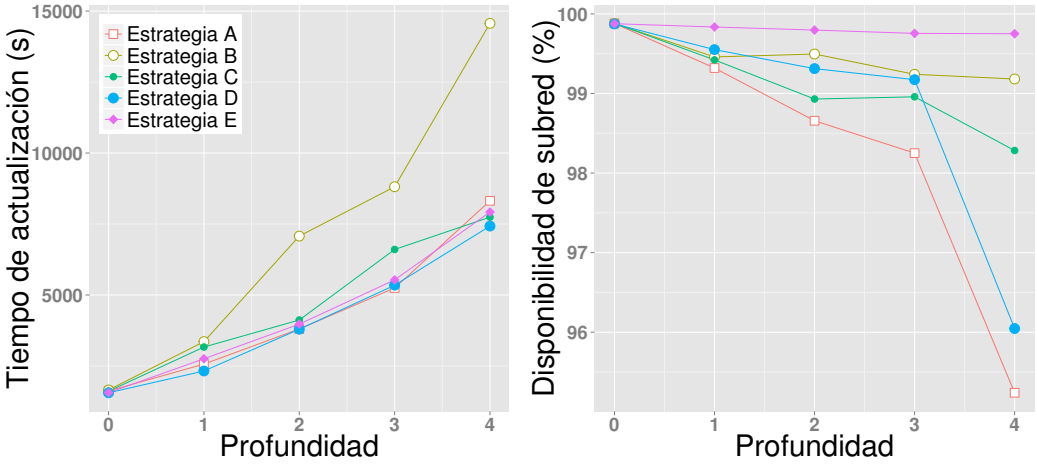
(a) Tiempo de actualización (b) Disponibilidad

Figura 5.15: Resultados en el escenario *Rural* de anchura 2



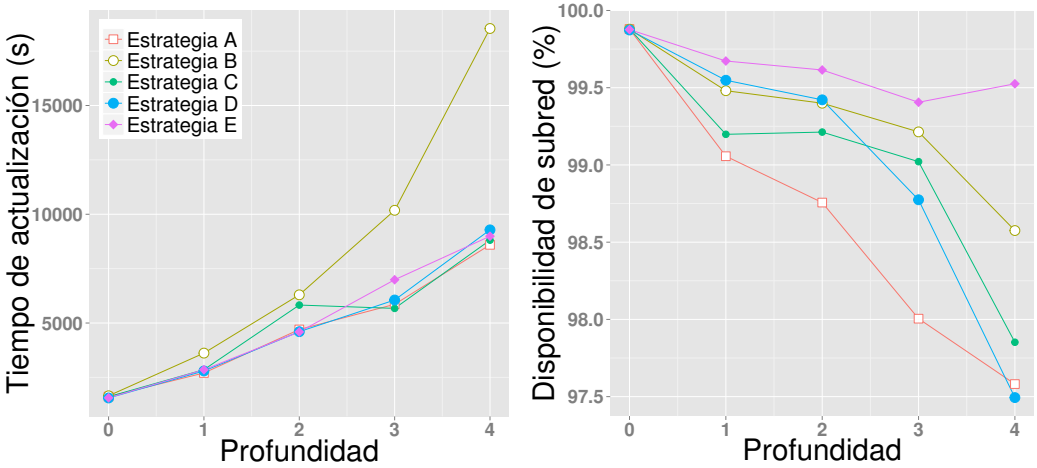
(a) Tiempo de actualización (b) Disponibilidad

Figura 5.16: Resultados en el escenario *Rural* de anchura 3



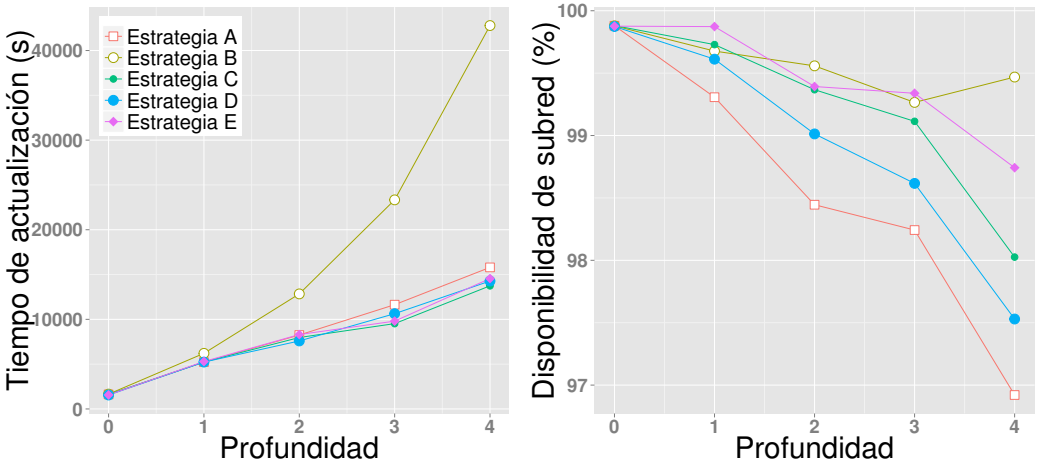
(a) Tiempo de actualización (b) Disponibilidad

Figura 5.17: Resultados en el escenario *Resindecial tipo 1* de anchura 1



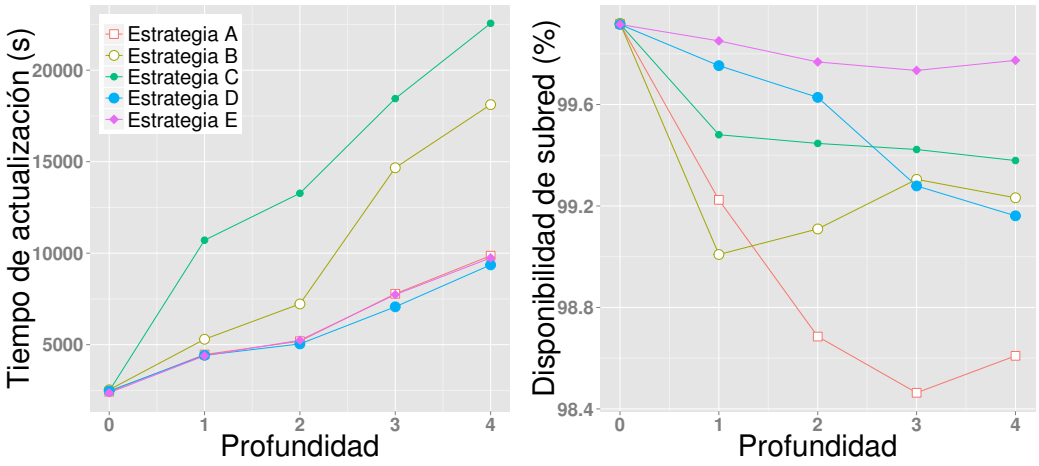
(a) Tiempo de actualización (b) Disponibilidad

Figura 5.18: Resultados en el escenario *Resindecial tipo 1* de anchura 2



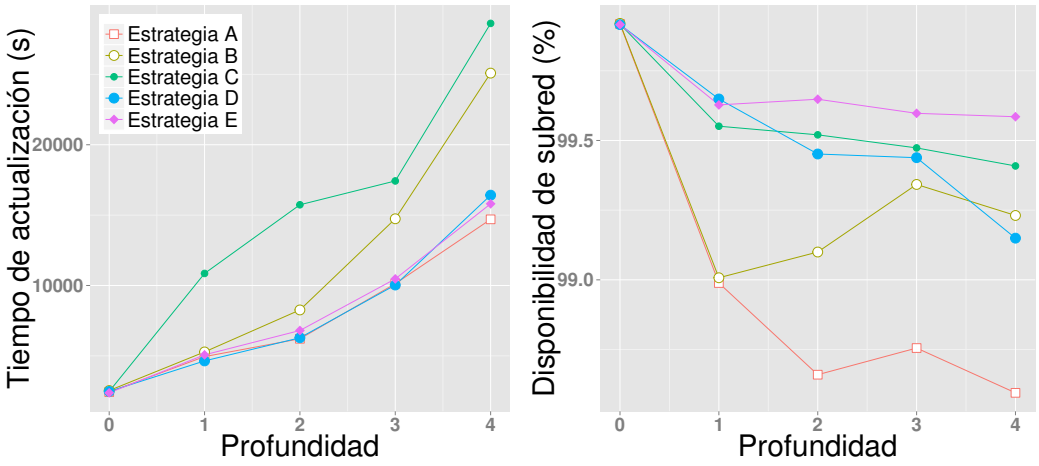
(a) Tiempo de actualización (b) Disponibilidad

Figura 5.19: Resultados en el escenario *Resindecial tipo 1* de anchura 4



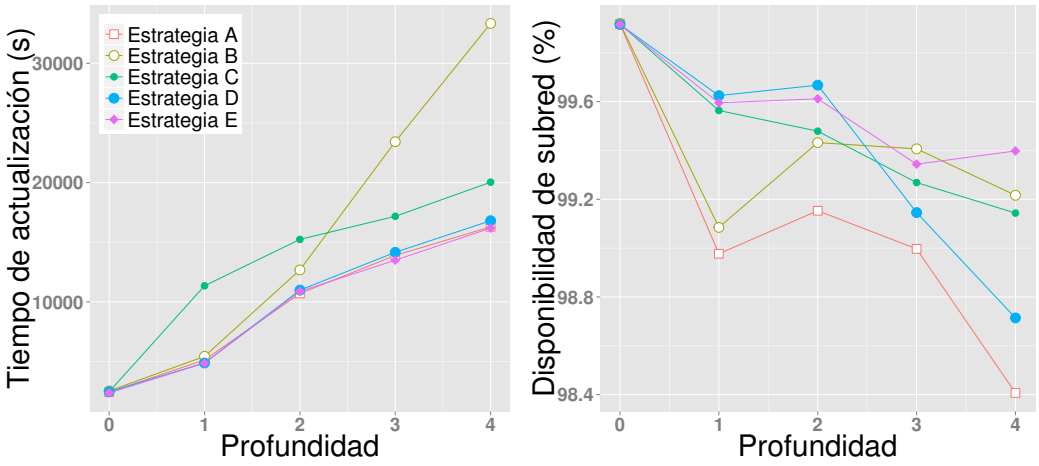
(a) Tiempo de actualización (b) Disponibilidad

Figura 5.20: Resultados en el escenario *Resindecial tipo 2* de anchura 1



(a) Tiempo de actualización (b) Disponibilidad

Figura 5.21: Resultados en el escenario *Resindecial tipo 2* de anchura 2



(a) Tiempo de actualización (b) Disponibilidad

Figura 5.22: Resultados en el escenario *Resindecial tipo 2* de anchura 3

Tabla 5.4: Resultados de disponibilidad de subred medios (en el eje de profundidad)

Rural						
	Anchura 1		**Anchura 2**		**Anchura 3**	
Estrategia	Disp.(%)	Pts	Disp.(%)	Pts	Disp.(%)	Pts
A	97,55	1	96,32	1	96,64	1
B	99,33	3	98,88	3	98,63	3
C	99,35	4	98,91	4	99,14	4
D	98,56	2	97,69	2	97,83	2
E	99,71	5	99,62	5	99,37	5
Residencial tipo 1						
	Anchura 1		**Anchura 2**		**Anchura 4**	
Estrategia	Disp.(%)	Pts	Disp.(%)	Pts	Disp.(%)	Pts
A	98,27	1	98,65	1	98,56	1
B	99,45	4	99,31	4	99,57	5
C	99,09	3	99,03	3	99,22	3
D	98,79	2	99,02	2	98,93	2
E	99,80	5	99,61	5	99,44	4
Residencial tipo 2						
	Anchura 1		**Anchura 2**		**Anchura 3**	
Estrategia	Disp.(%)	Pts	Disp.(%)	Pts	Disp.(%)	Pts
A	98,98	1	98,98	1	99,09	1
B	99,32	2	99,32	2	99,41	2
C	99,53	3	99,57	4	99,47	4
D	99,55	4	99,52	3	99,42	3
E	99,81	5	99,67	5	99,57	5

de disponibilidad, teniendo en cuenta que hay 5 estrategias, la estrategia con el valor más alto obtendrá 5 puntos, mientras que la estrategia con el valor más bajo solamente logrará 1 punto. En el caso del tiempo de actualización, un valor pequeño significa que el proceso de actualización del *firmware* tarda menos tiempo por lo que los valores de tiempo más altos obtendrán las puntuaciones más bajas. Por último, la tabla 5.6 muestra un resumen de los puntos obtenidos por las distintas estrategias en cada escenario, así como la clasificación final.

5.4.2 Análisis global de los resultados

Como ya se ha adelantado, en el apartado anterior, los gráficos muestran la tendencia de los resultados de las diferentes estrategias a medida que la profundidad y la anchura de las subredes crece, es decir, a medida que la topología lógica de los escenarios se vuelve más compleja. Las tablas 5.4, 5.5 y 5.6, en cambio, facilitan el análisis de los resultados que son más difíciles de discernir en los gráficos.

Como se puede observar en los gráficos, en las subredes menos profundas, las diferencias entre las estrategias son prácticamente despreciables, tanto en la disponibilidad de subred como en los resultados de tiempo de actualización. Sin embargo, a medida que la profundidad de las subredes aumenta estas diferencias crecen, especialmente en el caso de la disponibilidad de subred. Además, a medida que la topología lógica de la red se vuelve más compleja, los resultados son peores, es decir, en el caso de la disponibilidad los valores son más bajos, mientras que el tiempo de actualización aumenta.

Una profundidad de subred más alta significa que los paquetes necesitan más saltos para llegar a los nodos de servicio más lejanos del nodo base, de modo que generalmente el tiempo de actualización aumenta. Por otra parte, debido a que hay más niveles de jerarquía, también habrá más nodos de servicio que dependan de otros, por lo que cuando los *Switches* más cercanos al nodo base se reinician se verán afectados un mayor número de nodos de servicio.

En el caso de la anchura, un valor más alto significa que hay más nodos de tipo *Switch* en la subred (hay más ramas) por lo que hay una mayor probabilidad de que ocurran colisiones entre paquetes. Esto es debido a que en las estrategias A, C, D y E todos los nodos de servicio de la red pertenecen al mismo grupo *multicast* mientras que en la estrategia B los grupos *multicast* se definen nivel a nivel, y por lo tanto, en todas las estrategias los hijos directos de los nodos *Switch* pertenecen al mismo grupo *multicast*. Los nodos *Switch* del mismo nivel reciben las páginas del *firmware* con una separación temporal muy pequeña y como sus hijos pertenecen al mismo grupo *multicast*, todos ellos realizan la conmutación de paquetes que se dirigen al siguiente nivel en la jerarquía. Como resultado, la probabilidad de colisiones entre paquetes aumenta debido al intento simultáneo de acceso al canal de los nodos *Switch*.

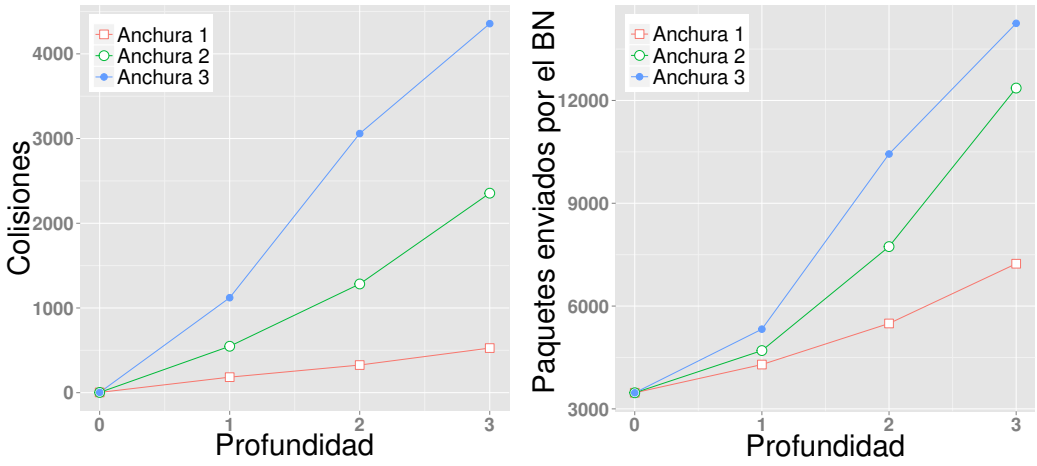
Del mismo modo, las colisiones también pueden ocurrir entre paquetes enviados por *Switches* de distinto nivel, por lo tanto, a medida que crece la profundidad de la red también aumenta la probabilidad de que ocurran colisiones. Debido a un mayor número de colisiones, el nodo de base tendrá que retransmitir un mayor número de paquetes (de datos y de control) por lo que el proceso de actualización se alargará en el tiempo. La disponibilidad de subred también puede verse afectada por el aumento de las colisiones ya que los paquetes de control que toman parte en el proceso de registro de los nodos de servicio y en el proceso *Keep-Alive* también pueden sufrir colisiones. En consecuencia, los

Tabla 5.5: Resultados de tiempo de actualización medios (en el eje de profundidad)

Rural						
	Anchura 1		**Anchura 2**		**Anchura 3**	
Estrategia	Tiempo(s)	Pts	Tiempo(s)	Pts	Tiempo(s)	Pts
A	1772,51	2	2485,99	2	2872,96	3
B	3279,55	1	4839,57	1	4934,48	1
C	1506,22	5	2211,51	5	2724,27	5
D	1669,16	3	2360,45	4	2919,31	2
E	1596,91	4	2435,22	3	2749,57	4
Residencial tipo 1						
	Anchura 1		**Anchura 2**		**Anchura 4**	
Estrategia	Tiempo(s)	Pts	Tiempo(s)	Pts	Tiempo(s)	Pts
A	4310,18	4	4696,04	5	8507,29	2
B	7093,43	1	8057,99	1	17363,87	1
C	4648,47	2	4955,07	3	7617,12	5
D	4088,91	5	4858,71	4	7856,57	4
E	4347,91	3	4997,75	2	7903,15	3
Residencial tipo 2						
	Anchura 1		**Anchura 2**		**Anchura 3**	
Estrategia	Tiempo(s)	Pts	Tiempo(s)	Pts	Tiempo(s)	Pts
A	5945,01	3	7683,332	5	9690,63	4
B	9568,61	2	11180,93	2	15483,34	1
C	13492,34	1	15022,58	1	13253.07	2
D	5672,69	5	7969,47	4	9859,67	3
E	5896,87	4	8103,51	3	9560,19	5

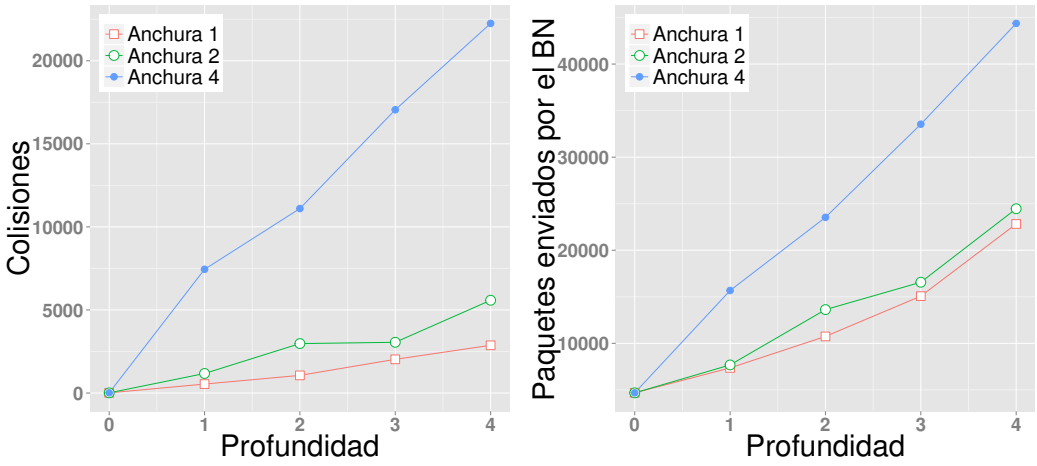
nodos de servicio pueden pasar más tiempo en el estado funcional de *Desconectado* (no disponible) al encontrarse con mayores dificultades a la hora de volver registrarse en la subred después de su reinicio o del de sus padres.

Los gráficos de las figuras que van desde la 5.23 a la 5.37 muestran la influencia que tienen los parámetros de anchura y profundidad sobre las colisiones y, en consecuencia, en el número de paquetes retransmitidos por el nodo base.



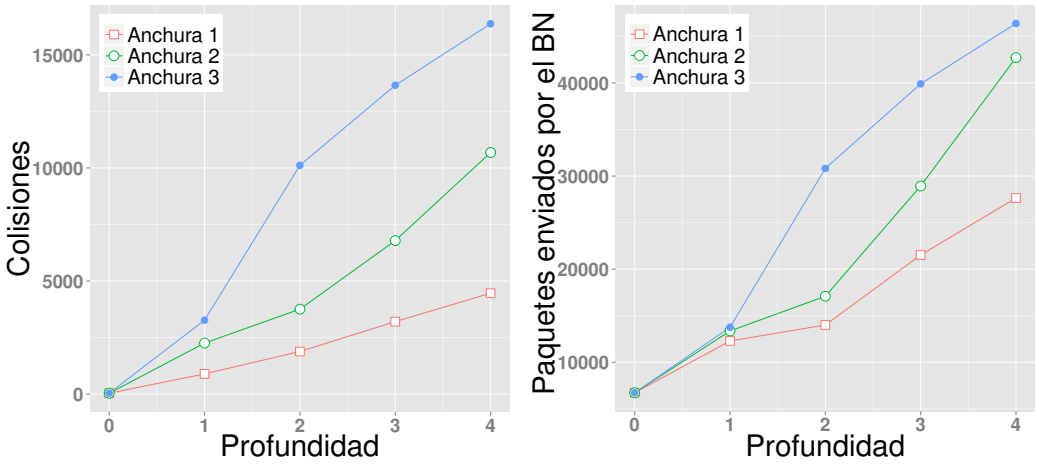
(a) Colisiones (b) Retransmisiones

Figura 5.23: Resultados de la estrategia A en el escenario *Rural*



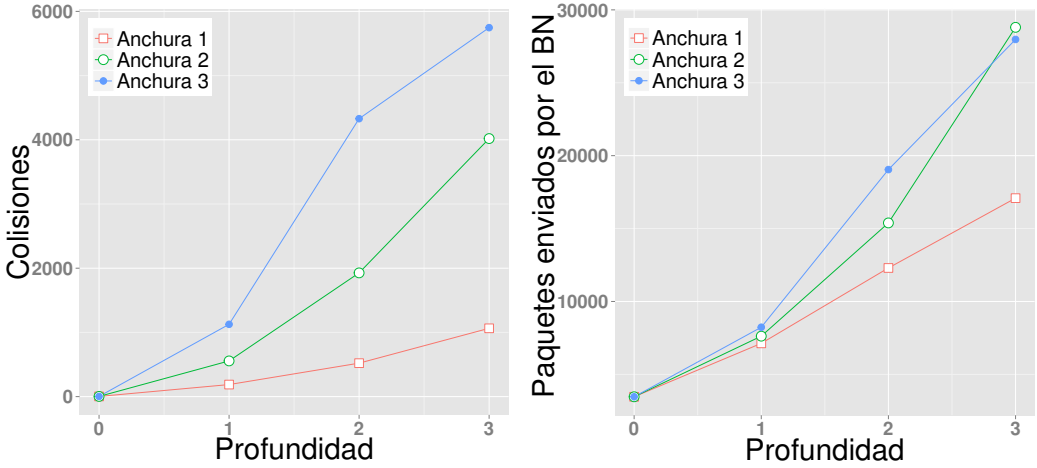
(a) Colisiones (b) Retransmisiones

Figura 5.24: Resultados de la estrategia A en el escenario *Residencial tipo 1*



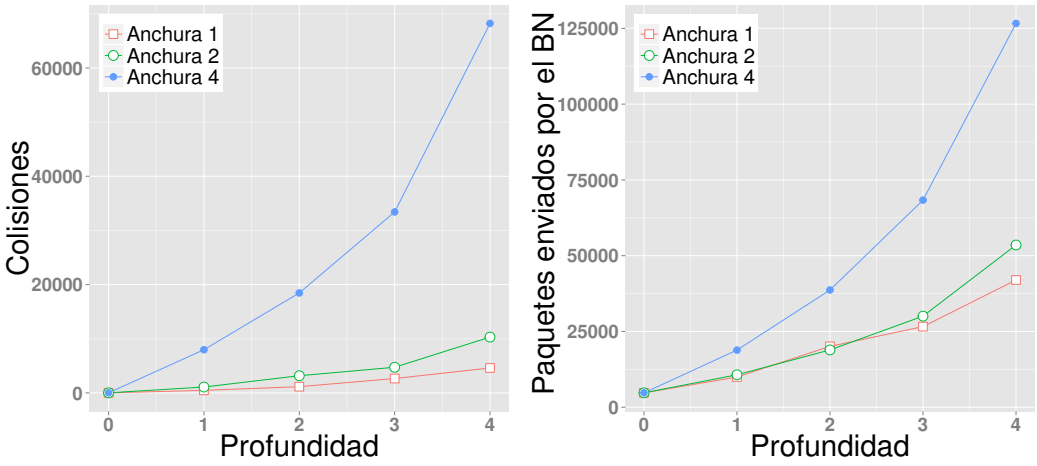
(a) Colisiones (b) Retransmisiones

Figura 5.25: Resultados de la estrategia A en el escenario *Residencial tipo 2*



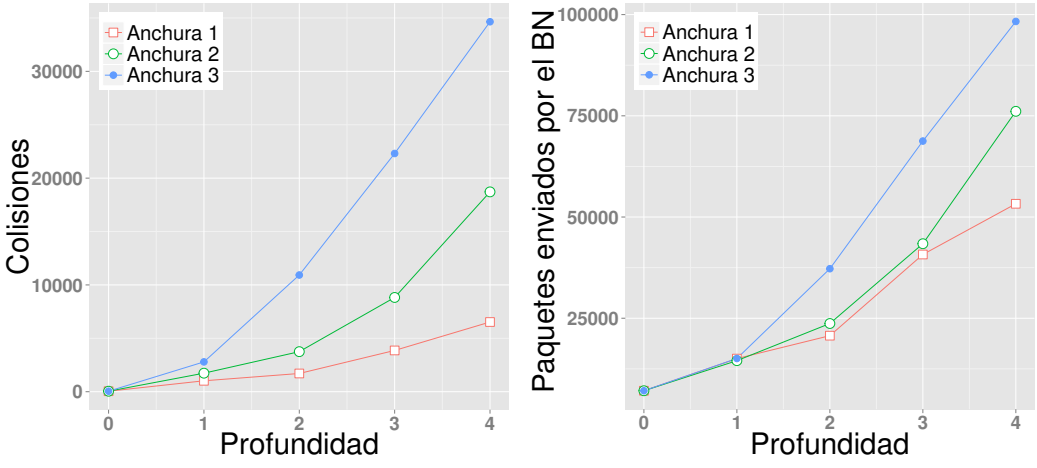
(a) Colisiones (b) Retransmisiones

Figura 5.26: Resultados de la estrategia B en el escenario *Rural*



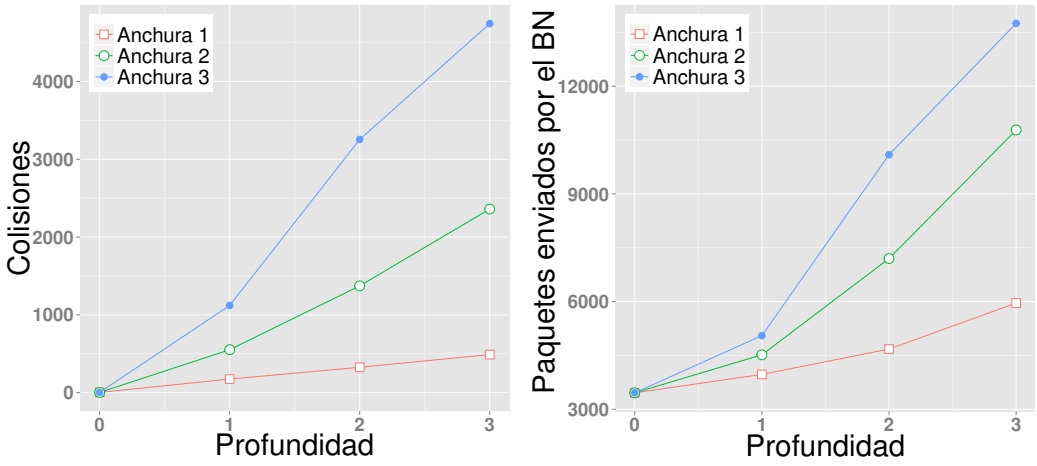
(a) Colisiones (b) Retransmisiones

Figura 5.27: Resultados de la estrategia B en el escenario *Residencial tipo 1*



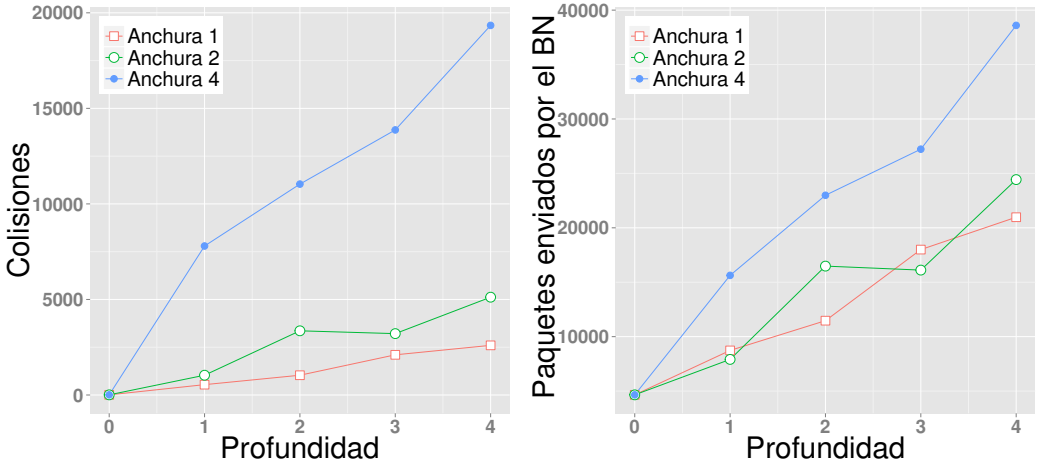
(a) Colisiones (b) Retransmisiones

Figura 5.28: Resultados de la estrategia B en el escenario *Residencial tipo 2*



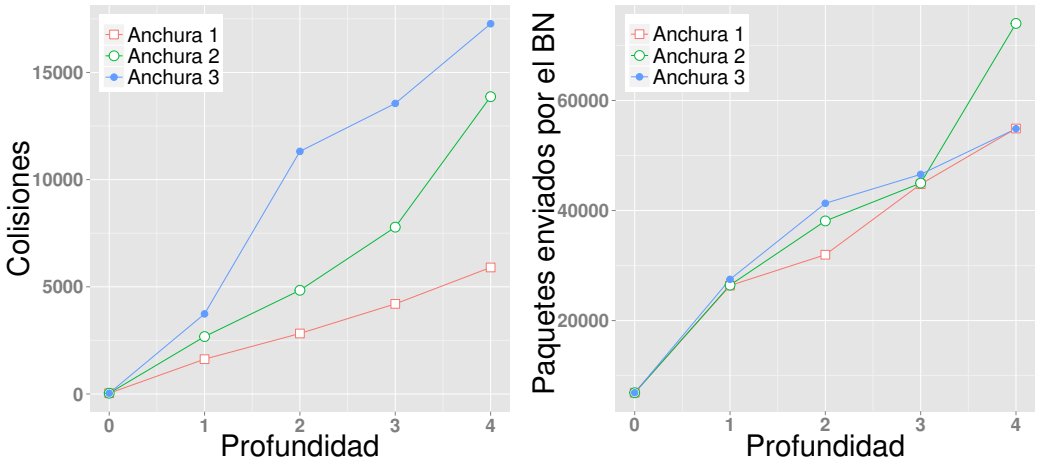
(a) Colisiones (b) Retransmisiones

Figura 5.29: Resultados de la estrategia C en el escenario *Rural*



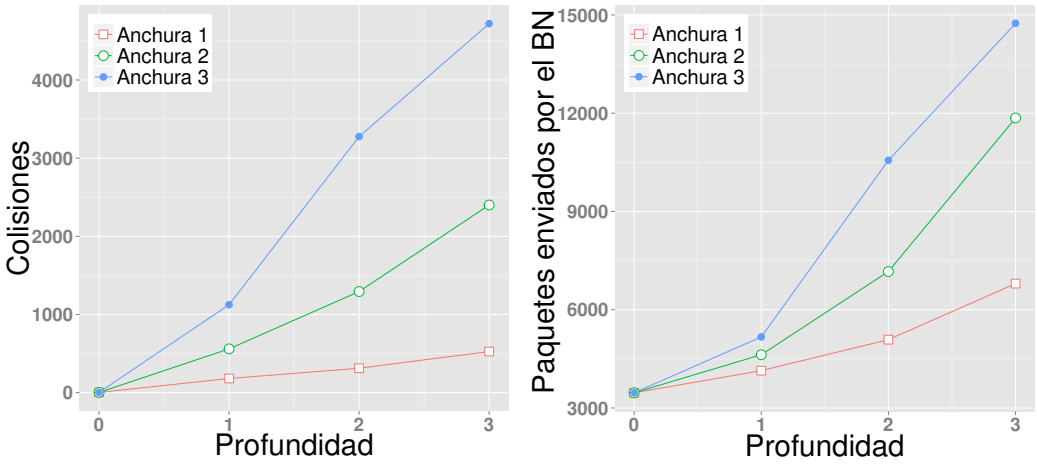
(a) Colisiones (b) Retransmisiones

Figura 5.30: Resultados de la estrategia C en el escenario *Residencial tipo 1*



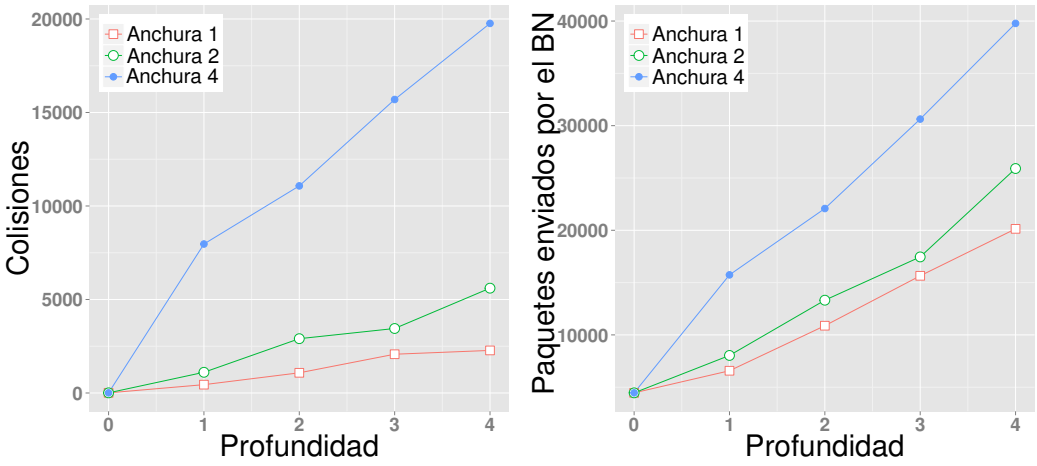
(a) Colisiones (b) Retransmisiones

Figura 5.31: Resultados de la estrategia C en el escenario *Residencial tipo 2*



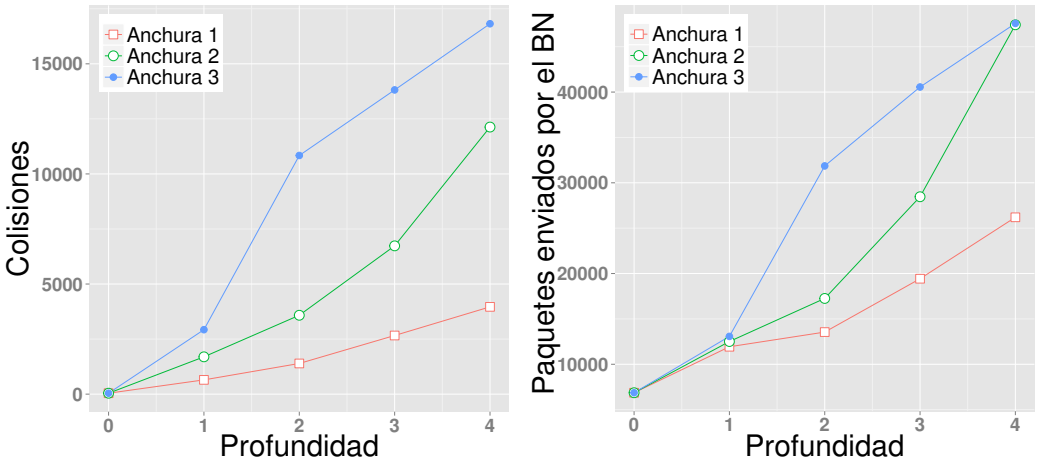
(a) Colisiones (b) Retransmisiones

Figura 5.32: Resultados de la estrategia D en el escenario *Rural*



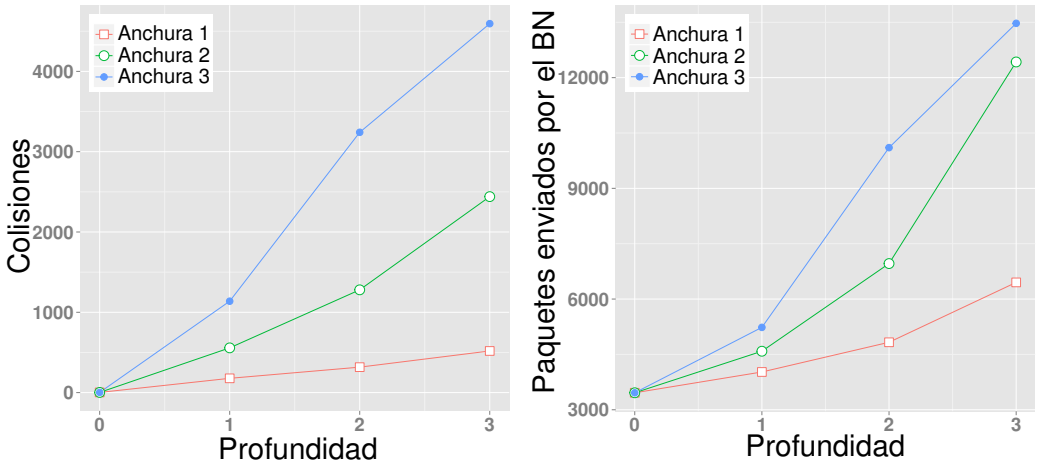
(a) Colisiones (b) Retransmisiones

Figura 5.33: Resultados de la estrategia D en el escenario *Residencial tipo 1*



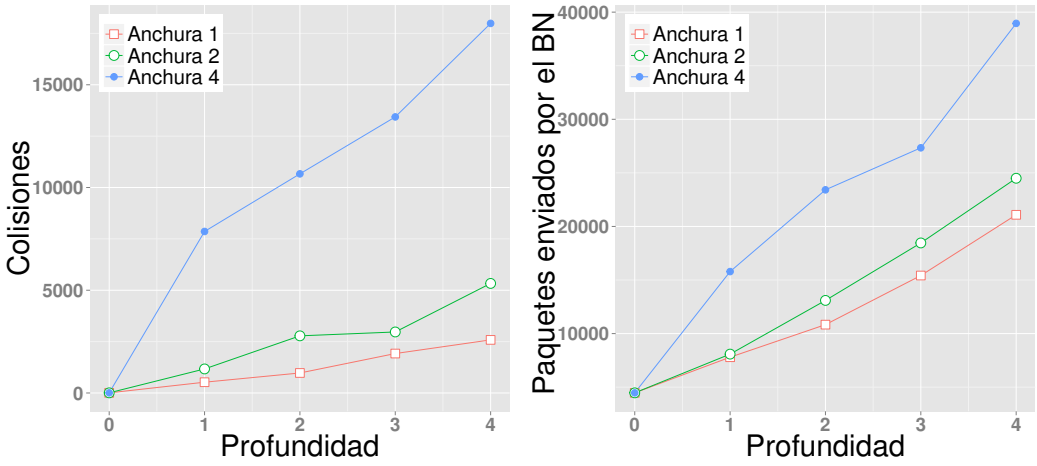
(a) Colisiones (b) Retransmisiones

Figura 5.34: Resultados de la estrategia D en el escenario *Residencial tipo 2*



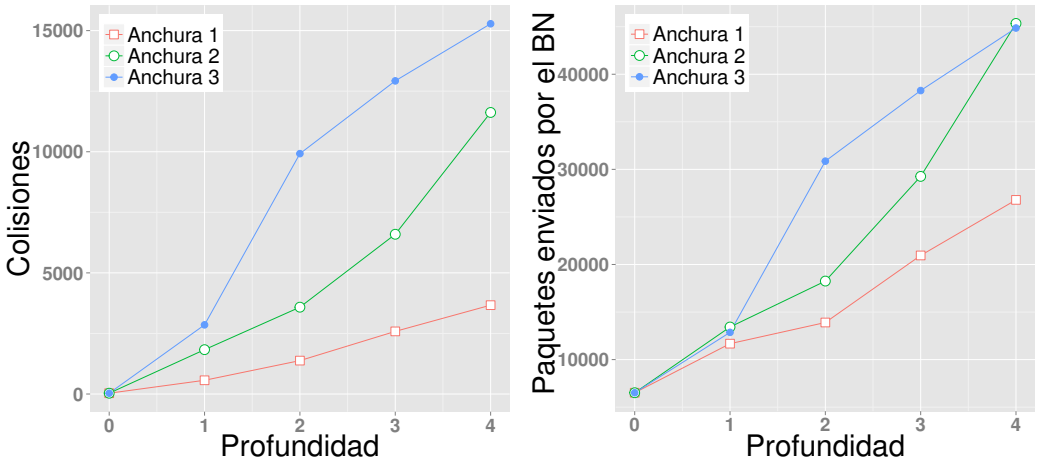
(a) Colisiones (b) Retransmisiones

Figura 5.35: Resultados de la estrategia E en el escenario *Rural*



(a) Colisiones (b) Retransmisiones

Figura 5.36: Resultados de la estrategia E en el escenario *Residencial tipo 1*



(a) Colisiones (b) Retransmisiones

Figura 5.37: Resultados de la estrategia E en el escenario *Residencial tipo 2*

Tabla 5.6: Clasificación de los resultados de tiempo y disponibilidad

Disponibilidad								
	Rural		**Residencial 1**		**Residencial 2**		**TOTAL**	
Estrategia	Pts	Posición	Pts	Posición	Pts	Posición	**Pts**	**Posición**
A	3	5	3	5	3	5	**9**	**5**
B	9	3	13	2	6	4	**28**	**3**
C	12	2	9	3	11	2	**32**	**2**
D	6	4	6	4	10	3	**22**	**4**
E	15	1	14	1	15	1	**44**	**1**
Tiempo								
	Rural		**Residencial 1**		**Residencial 2**		**TOTAL**	
Estrategia	Pts	Posición	Pts	Posición	Pts	Posición	**Pts**	**Posición**
A	7	4	11	2	12	1	**30**	**3**
B	3	5	3	5	5	2	**11**	**5**
C	15	1	10	3	4	3	**29**	**4**
D	9	3	13	1	12	1	**34**	**1**
E	11	2	8	4	12	1	**31**	**2**

5.4.3 Discusión comparativa de los resultados

Tomando en cuenta todas las consideraciones anteriores, la estrategia E es la que ofrece los mejores resultados en términos de disponibilidad de subred para cada uno de los escenarios que se presentan en este trabajo. Como se puede observar en los gráficos, esta estrategia es la que mejor escala a medida que la profundidad y la anchura de la subred aumentan, es decir, la disponibilidad se degrada en menor medida que en el resto de las estrategias. Del mismo modo, la estrategia A es la que muestra los peores resultados de disponibilidad de subred y la que peor evoluciona a medida que la topología lógica de la red se vuelve más compleja. Además, en las subredes más profundas, donde se esperan los peores resultados, la estrategia E mejora la disponibilidad de subred hasta en 7 puntos porcentuales con respecto a la peor estrategia que es la A. Dicha mejora de la disponibilidad puede ser debida a que en la estrategia E el *firmware* de los nodos *Switch* se activa después de activar el *firmware* de sus hijos, por lo que los hijos serán reiniciados en primer lugar haciendo que la disponibilidad de subred mejore. Por otro lado, en la clasificación final, la estrategia E tiene los segundos mejores resultados en términos de tiempo de actualización, por lo que ambas

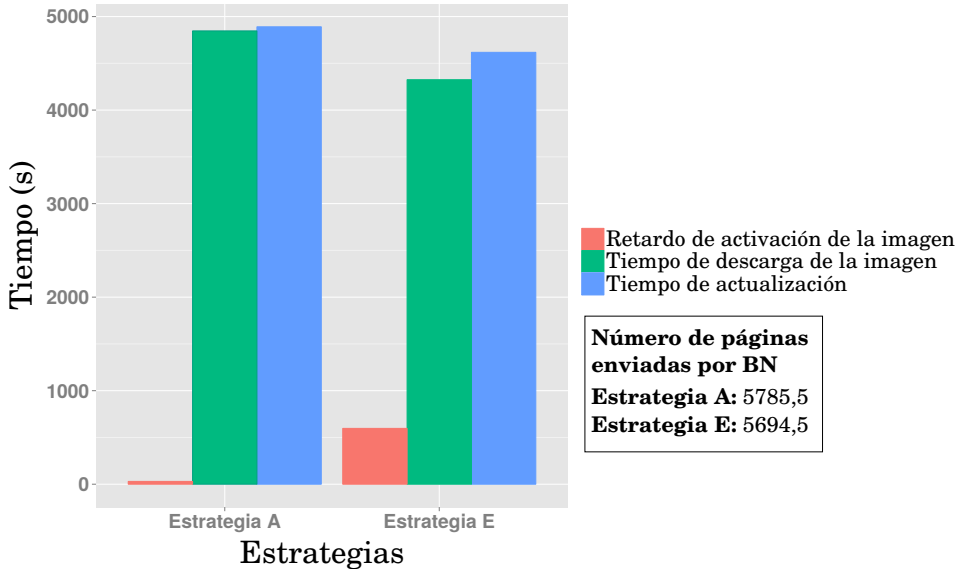


Figura 5.38: Comparación de las estrategias A y E en términos de tiempo de actualización, tiempo de descarga de la imagen y retardo de activación en el escenario *Rural* de anchura 3 y profundidad 3.

métricas son mejores que en la estrategia A. Sin embargo, la mejora observada en el tiempo de actualización con respecto a la estrategia A es menos importante, aunque como se esperaba la estrategia E no empeora el tiempo de actualización global. Esto se debe a que una mejor disponibilidad de la red se traduce en que la fase de descarga de la imagen del *firmware* del total de los nodos que componen la red concluya antes, ya que el nodo base tendrá que reenviar un menor número de páginas del *firmware*, ahorrando así una importante cantidad de tiempo. Sin embargo, esta rapidez en la fase de descarga se contrarresta en cierta medida por un retardo medio mayor en la activación de la imagen de cada nodo de servicio (mayor tiempo de espera desde que un nodo de servicio recibe la imagen hasta que ésta es activada), tal y como muestran las figuras 5.38, 5.39 y 5.40, debido a que los nodos en estado *Switch* tendrán que esperar a que todos sus hijos hayan sido actualizados. Por esta razón, aunque la estrategia E es más rápida que la A, dichas diferencias no son tan significativas.

La estrategia C es la estrategia que muestra los segundos mejores resultados en términos de disponibilidad de subred. Sin embargo, esta estrategia presenta peores resultados en cuanto a tiempo de actualización si los comparamos con la estrategia A especialmente en las grandes redes como en los escenarios *Residencial tipo 1* y *Residencial tipo 2*. En esta estrategia, el nodo base espera hasta que todos los nodos de servicio inicializados hayan recibido la imagen del *firmware*

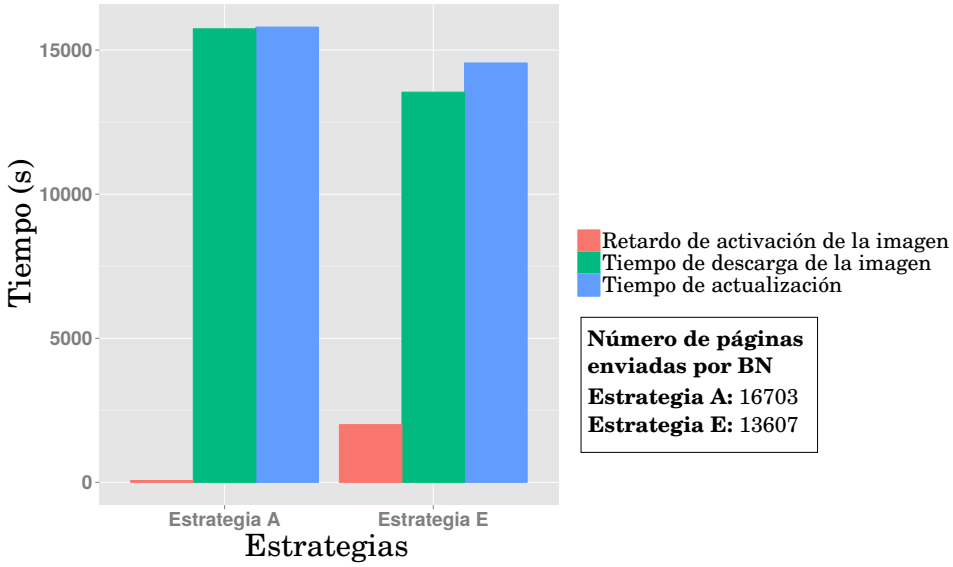


Figura 5.39: Comparación de las estrategias A y E en términos de tiempo de actualización, tiempo de descarga de la imagen y retardo de activación en el escenario *Residencial tipo 1* de anchura 4 y profundidad 4.

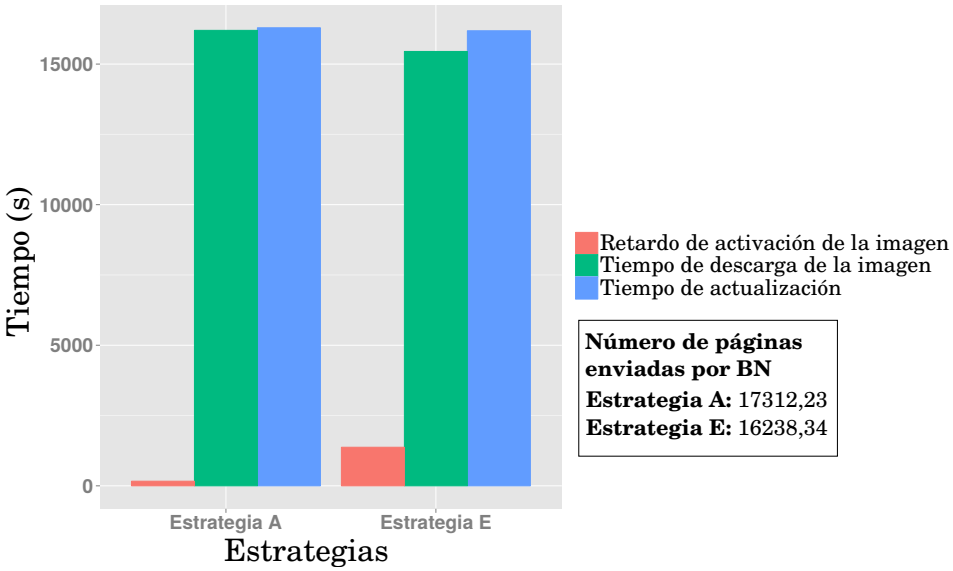


Figura 5.40: Comparación de las estrategias A y E en términos de tiempo de actualización, tiempo de descarga de la imagen y retardo de activación en el escenario *Residencial tipo 2* de anchura 3 y profundidad 4.

para comenzar con la fase de activación, por lo que el retardo de activación de la imagen de los nodos también es elevado si lo comparamos con la estrategia A. Por otro lado, a la hora de realizar dicha activación, en primer lugar el nodo base ordena los identificadores de los nodos de servicio en función de su nivel en la jerarquía en orden descendente. De esta manera, en primer lugar, el nodo base activará el *firmware* de los nodos de servicio que están más lejos del nodo base y continuará con los más cercanos. Sin embargo, si la red es grande, es más probable que sea necesaria más de una ronda para actualizar todos los nodos de servicio de la subred. Así que, aunque al comienzo de la fase de activación los nodos de servicio se ordenan en función de su nivel jerárquico en forma descendente, puede que algunos de los *Switches* se reinicien antes que sus hijos (los cuales están a la espera para la siguiente ronda) y en consecuencia, la activación no se realiza en orden descendente lo que hace que la disponibilidad de la red empeore.

La tercera mejor estrategia en términos de disponibilidad de subred es la estrategia B. En esta estrategia, como en la estrategia C, los resultados empeoran en los escenarios más grandes como es el caso del escenario *Residencial tipo 2*. En este tipo de escenarios, cuando el nodo base está tratando de actualizar el grupo de *multicast* que incluye a los nodos de servicio más alejados del nodo base, si no logra actualizar todos los nodos de dicho grupo después de un número de intentos predefinido, el nodo base tratará de actualizar los nodos de servicio del siguiente grupo de *multicast* de la lista (con un nivel menos en la jerarquía). Por lo tanto, es posible que antes de que se actualicen todos los nodos de servicio de un nivel concreto, los nodos que actúan como *Switch* de estos nodos sean actualizados antes de lo esperado, y debido a que el orden de actualización se ve alterado, la disponibilidad de subred empeora. Sin embargo, aunque esta estrategia presenta mejores resultados de disponibilidad subred que la estrategia A, esta estrategia tiene los peores resultados en tiempo de actualización de todas las estrategias. Esto puede ser debido al hecho de que en esta estrategia se define un grupo *multicast* para cada nivel de jerarquía, por lo que la imagen de *firmware* se envía por completo a cada uno de los grupos por separado. Esto da lugar a una ocupación de canal ineficiente, ya que muchas de las páginas del *firmware* que se dirigen a las partes más profundas de la subred (para grupos *multicast* con los niveles de jerarquía más altos) son descartadas o no son almacenadas por los nodos de servicio más cercanos al nodo base. Por lo tanto, esto implica que el nodo base tenga que retransmitir el mismo paquete de datos (página del *firmware*) para cada uno de los grupos de *multicast*.

La estrategia D es la cuarta mejor estrategia en términos de disponibilidad de subred, sin embargo, es la estrategia con los mejores resultados en tiempo de actualización. Esta estrategia intenta dar prioridad a la activación de los nodos *Terminales*, por lo que cuando un nodo de servicio notifica al nodo base que ya ha recibido las páginas del *firmware* correctamente, el nodo base comprueba

si el nodo de servicio es un *Terminal* o un *Switch*. Si es un nodo *Terminal*, el nodo base activará el *firmware* del nodo, mientras que si se trata de un nodo *Switch*, el nodo base esperará hasta que todos los nodos *Terminales* inicializados hayan sido activados antes de proceder con su activación. Sin embargo, cuando finalmente el nodo base comienza con la activación de los nodos *Switch*, no se tiene en cuenta si estos nodos dependen unos de otros en la jerarquía. Además, si algunos de los nodos de servicio se han quedado a la espera de ser actualizados en las siguientes rondas de actualización, es posible que algunos de los nodos *Switch* sean activados antes que otros nodos *Terminales* que a su vez dependan de ellos. Por otra parte, esta estrategia es más rápida que la estrategia E ya que en este caso los nodos *Switch* no tienen que esperar hasta que sus hijos sean actualizados (el retardo medio de activación de la imagen es menor), de modo que aunque representa un empeoramiento de la disponibilidad de subred supone cierto ahorro de tiempo.

Por último, la estrategia A es la peor de todas las estrategias en términos de disponibilidad de subred. Esto puede ser debido a que en la estrategia A la activación del *firmware* de los nodos no tiene en cuenta ningún aspecto lógico de la topología, y por lo tanto, el *firmware* de los nodos de servicio se activa de forma aleatoria tan pronto como notifiquen al nodo base que han recibido la imagen del *firmware* correctamente. Además, teniendo en cuenta que es muy probable que los nodos de servicio más cercanos al nodo base sean los primeros en recibir la imagen, el *firmware* de estos nodos también será activado en primer lugar lo que implica un empeoramiento significativo de la disponibilidad de subred. Por otro lado, los resultados de tiempo de actualización de esta estrategia también son peores que en las estrategias D y E, aunque las diferencias observadas en este aspecto son menos significativas si las comparamos con las de la disponibilidad de subred.

Capítulo

6

Conclusiones

Este capítulo recoge las contribuciones principales realizadas en este trabajo de investigación así como las posibles líneas futuras de investigación que pueden continuar este trabajo. El capítulo se encuentra estructurado de la siguiente manera: la sección 6.1 proporciona un resumen de las contribuciones técnicas y científicas realizadas, mientras que la sección 6.2 ofrece una visión de las posibles direcciones futuras de investigación.

6.1 Contribuciones

El objetivo general de este trabajo de investigación era mejorar la disponibilidad de las subredes PRIME durante el proceso de actualización *firmware*. Además, teniendo en cuenta que el proceso de actualización *firmware* se caracteriza por ser un proceso de larga duración, esta mejora de la disponibilidad no debe suponer un aumento del tiempo necesario para realizar la actualización de los nodos de servicio de la red.

La hipótesis de partida formulada en base a este objetivo mantiene que *la disponibilidad de las subredes PRIME durante el proceso de actualización firmware de los nodos de servicio que componen dicha subred puede ser mejorada sin aumentar la duración total del proceso mediante el diseño de estrategias de actualización que tomen en cuenta la topología lógica de la subred.*

Para la consecución del objetivo general y, por tanto, para llevar a cabo la validación de la hipótesis planteada fueron establecidos dos objetivos específicos:1) *desarrollar y validar un modelo de simulación de subredes PRIME que permita evaluar la aplicación de actualización firmware* y 2) *diseñar, implementar y evaluar distintas estrategias de actualización firmware que mejoren la disponibilidad de la subred sin que a cambio suponga un aumento del tiempo total necesario*

para llevar a cabo la actualización. Es importante subrayar que los objetivos específicos definidos que propiciaban el cumplimiento el objetivo general de este trabajo han sido logrados satisfactoriamente y que se ha validado la hipótesis planteada. El cumplimiento de dichos objetivos específicos ha dado lugar a las siguientes contribuciones técnicas y científicas.

En el *capítulo 4* se describe la principal contribución técnica de este trabajo: el modelo de simulación de subredes PRIME. Dicho modelo de simulación se ha desarrollado siguiendo la versión 1.3.6 del estándar PRIME y se ha validado con resultados de pruebas realizadas sobre redes reales, por lo que se trata de una herramienta de gran utilidad para el diseño y evaluación de nuevas aplicaciones y algoritmos antes de su aplicación práctica. Por otra parte, debido a que ciertos aspectos del estándar son de libre implementación y cada fabricante los implementa de forma distinta, se ha llevado a cabo el diseño e implementación de los siguientes algoritmos que a su vez, constituyen una aportación:

- *Algoritmo de promoción basado en la topología lógica más común*. El modelo de simulación emplea un algoritmo de promoción basado en un fichero XML que contiene la topología lógica más común de la red a simular. Cuando el nodo base recibe solicitudes de promoción, para escoger qué nodo de servicio ha de ser promocionado, consulta el fichero mencionado, y en función del origen de la petición, escoge el nodo de servicio que pasará del estado *Terminal* al de *Switch* (ver apartado 4.2.3 y 4.2.5.2 del capítulo 4).

- *Algoritmo de asignación de slot de balizas*. Cuando un nodo de servicio cambia su estado de *Terminal* a *Switch*, el nodo base tendrá que indicarle al nuevo *Switch* el *slot* en el que deberá transmitir su baliza. El nodo base deberá reservar un *beacon-slot* de la trama para la emisión de las balizas de dicho *Switch*. El algoritmo de asignación de balizas no se detalla en el estándar, por lo que se ha implementado un algoritmo de asignación propio (ver 4.2.5.2 del capítulo 4).

- *Algoritmo de elección de frecuencia de envío de paquetes Keep-Alive*. El criterio seguido por el nodo base para escoger los valores de los *REG.TIME*, *ALV.TIME* o *PRO.TIME*, así como la frecuencia de envío de paquetes *Keep-Alive* no está especificado en el estándar, por lo que cada fabricante escoge el suyo. En el modelo de simulación desarrollado, la asignación de estos valores se ha realizado mediante un algoritmo basado en el nivel de confianza de los nodos (ver 4.2.5.2 del capítulo 4).

- *Algoritmo de denegación de ayuda ante la recepción de paquetes PNPDU*. El algoritmo mediante el cual un nodo de servicio en el estado *Terminal* toma la decisión de atender o no a un paquete PNPDU enviado por un nodo de servicio en estado *Desconectado* no se concreta en el estándar y

cada fabricante lo implementa como considera oportuno. En el modelo de simulación, el algoritmo implementado se basa en el parámetro *ignorePNPDUpercentage* definido en el fichero de configuración del modelo de simulación. Este parámetro sirve para fijar el porcentaje de paquetes *PNPDU* a ignorar por los nodos en estado *Terminal* (ver 4.2.5.2 del capítulo 4).

Por otra parte, teniendo en cuenta que en el modelo de simulación no existe un medio físico real, para simular el mecanismo de comprobación de canal llevado a cabo por equipos reales, y de fenómenos como la reproducción de colisiones y la pérdida de paquetes por la degradación SNR de la línea, se han diseñado e implementado los siguientes algoritmos que son también una aportación de este trabajo de investigación.

- *Algoritmo de comprobación de canal*. Cuando un nodo ejecuta el mecanismo CSMA/CA de acceso al medio realiza una o varias comprobaciones de canal para saber si puede transmitir un paquete. Como en el modelo de simulación no existe un medio físico real como tal, el nodo de servicio le consulta el estado del canal al nodo *"Network Manager"*. Cuando un nodo transmite un paquete a la línea, envía una notificación al *"Network Manager"* y éste almacena la información del paquete en un vector durante el tiempo que dura su transmisión. Si durante el tiempo que el paquete se encuentra en el vector otro nodo realiza una consulta de canal, el *"Network Manager"* le indicará si el canal está ocupado o no, en función de si los nodos transmisores pueden escucharse entre sí; es decir, dependiendo de la diferencia máxima de niveles entre ellos, calculado a partir del parámetro *collisionDomain* del fichero de configuración (ver 4.2.6.1 del capítulo 4).

- *Algoritmo de reproducción de colisiones*. Si durante el tiempo en el cual un paquete se encuentra almacenado en el vector de paquetes en transmisión otro nodo transmite un paquete, el módulo *"Network Manager"* determinará si ha ocurrido una colisión entre los paquetes enviados en función de la diferencia máxima de niveles existente entre los nodos transmisores, calculado a partir del parámetro *collisionDomain* del fichero de configuración (ver 4.2.6.2 del capítulo 4).

- *Algoritmo de pérdida de paquetes debida a la degradación SNR de la línea*. Para reproducir el efecto de la degradación SNR en la línea, se ha implementado un mecanismo de pérdida de paquetes aleatorio basado en el valor del parámetro *packetLoss* definido en el fichero de configuración del modelo de simulación (ver apartado 4.2.5.1 del capítulo 4).

En el *capítulo 5* se incluye la principal aportación científica de este trabajo de investigación, que consiste en el diseño, implementación y evaluación de estrategias de actualización *firmware* de subredes PRIME con el objetivo de mejorar la disponibilidad de la red durante el proceso sin que suponga un aumento de la duración del mismo.

Para el diseño de dichas estrategias, se ha aprovechado la flexibilidad proporcionada por el estándar (ver 5.2 del capítulo 5) y se ha puesto especial atención en la activación del *firmware*, ya que se considera que es la fase que mayor influencia puede tener sobre los resultados de disponibilidad de la subred. Una vez implementadas, se han definido redes de distintas topologías (físicas y lógicas) y se ha llevado a cabo la simulación de las estrategias sobre dichas redes, lo que ha permitido evaluar su comportamiento y extraer las siguientes conclusiones.

- Es preferible crear un único grupo *multicast* que incluya a todos los nodos de un mismo fabricante frente a crear distintos grupos por fabricante y nivel de jerarquía, ya que en el segundo caso, además de no apreciarse una mejora significativa de la disponibilidad de la red, el proceso se alarga demasiado en el tiempo. Esto puede ser debido a que cuando se define más de un grupo en la red la imagen del *firmware* se envía a cada grupo por separado. Esto da lugar a una ocupación de canal ineficiente, ya que muchas de las páginas *firmware* que se dirigen a las partes más profundas de la subred son descartadas o no son almacenadas por los nodos de servicio más cercanos al nodo base. Por lo tanto, esto implica que el nodo base tenga que retransmitir el mismo paquete de datos (página del *firmware*) para cada uno de los grupos de *multicast* y el proceso de actualización se alargue demasiado.

- A medida que aumenta la profundidad y la anchura los resultados de tiempo de actualización y disponibilidad de red empeoran.

  - *Tiempo de actualización*. Una profundidad de red más alta significa que los paquetes necesitan más saltos para llegar a los nodos de servicio más lejanos del nodo base, de modo que generalmente el tiempo de actualización aumenta. Una anchura de red más alta significa que hay más nodos *Switch* en la red por lo que hay una mayor probabilidad de que ocurran colisiones entre paquetes. Debido a un mayor número de colisiones, el nodo base tendrá que retransmitir un mayor número de paquetes (de datos y de control) por lo que el proceso de actualización tardará más tiempo. Las colisiones también aumentan a medida que crece la profundidad de la red puesto que las colisiones también ocurren entre paquetes enviados por *Switches* de distinto nivel.

  - *Disponibilidad de la red*. Una profundidad de red mayor significa que hay más niveles de jerarquía y habrá más nodos de servicio que dependan de otros, por lo que cuando los *Switches* más cercanos al nodo base se reinician afectará a un mayor número de nodos de servicio y la disponibilidad de la red empeorará. Por otro lado, una anchura y profundidad de red mayor implica que ocurran más colisiones, pudiendo afectar a los paquetes de control que toman parte en el proceso de registro de los nodos de servicio y en el proceso Keep-Alive. En consecuencia, los nodos de servicio pueden pasar más tiempo en el estado funcional *Desconectado* (no disponible) al encontrarse con más dificultades a la hora de volver a conectarse después de su reinicio o del de sus padres.

- En las subredes menos profundas las diferencias entre las estrategias estudiadas son prácticamente despreciables, tanto en la disponibilidad de subred como en los resultados de tiempo de actualización. Sin embargo, a medida que la profundidad de las subredes aumenta, estas diferencias crecen. Este trabajo propone una estrategia de actualización *firmware* que mejora claramente la disponibilidad de una subred PRIME (estrategia E) hasta en 7 puntos porcentuales en los escenarios más profundos sin empeorar el tiempo de actualización global, con respecto a la que peores resultados ofrece que, a su vez, fue empleada en la validación del modelo de simulación desarrollado (estrategia A). Además, dicha estrategia (estrategia E) es la que mejor escala a medida que la profundidad y la anchura de la subred aumentan; es decir, la disponibilidad se degrada en menor medida que en el resto de las estrategias. En ella, el *firmware* de los nodos *Switch* se activa después de activar el *firmware* de sus hijos, por lo que los hijos serán reiniciados antes que los padres haciendo que la disponibilidad de la subred sea mejor que en otras estrategias. En cambio, en la estrategia con peores resultados, no se tiene en cuenta ningún aspecto lógico de la topología y la activación del *firmware* de los nodos de servicio se realiza de forma aleatoria, tan pronto como estos le notifiquen al nodo base que han recibido la imagen del *firmware* de forma correcta.

- En las estrategias analizadas, una mayor disponibilidad de la subred durante el proceso de actualización *firmware* se traduce en una mayor rapidez en la fase de descarga de la imagen del *firmware*, ya que el nodo base tendrá que reenviar un número menor de páginas del *firmware* ahorrando así una importante cantidad de tiempo. Sin embargo, esta rapidez en la descarga puede verse contrarrestada por un mayor retardo en la activación de la imagen de cada nodo, haciendo que a pesar de obtener una mejora notable en términos de disponibilidad, dichas mejoras en términos de tiempo de actualización global de la red no sean tan significativas.

Por lo tanto, se ha demostrado que, tal y como establecía la hipótesis de partida, es posible mejorar la disponibilidad de la subred sin aumentar la duración total del proceso mediante el diseño de estrategias de actualización que tomen en cuenta la topología lógica de la subred.

6.2 Líneas futuras

En esta sección se recogen las posibles líneas de trabajo futuras, tanto para aspectos relacionados con el modelo de simulación de subredes PRIME como para el estudio de estrategias de actualización *firmware* de nodos de servicio de subredes PRIME.

6.2.1 Modelo de simulación de subredes PRIME

Durante el diseño e implementación del modelo de simulación de subredes PRIME, se han identificado las siguientes posibles mejoras a realizar:

- *Fichero XML con la topología lógica más común*. El modelo de simulación desarrollado emplea la topología lógica más común de la subred, obtenida a partir de un proceso de monitorización previo de la red real a simular. Actualmente, el procesamiento del fichero Excel que contiene la topología lógica más común proporcionada por el nodo base y la generación del fichero XML de entrada al modelo de simulación correspondiente (a partir del fichero Excel) se ha realizado de forma manual. Como trabajo futuro, esta prevista la implementación de un programa que se encargue de procesar y trasladar la información incluida en el fichero Excel al formato XML correspondiente, de forma que se pueda llevar a cabo la simulación masiva de escenarios reales de manera más rápida y fácil.

- *Algoritmo de asignación de balizas*. El algoritmo de asignación de balizas empleado en el modelo de simulación permite la reserva de hasta 64 *beacon-slots*. Aunque el tamaño de las redes simuladas en este trabajo no requiere la reserva de tantos *slots*, como trabajo futuro, se estudiará la opción de implementar un algoritmo que permita la asignación de un mayor número de *slots* de modo que se puedan simular redes de mayor tamaño.

6.2.2 Estudio de estrategias de actualización *firmware*

Durante el diseño e implementación de las estrategias de actualización *firmware* se han identificado también posibles líneas de investigación futuras que pueden dar como resultado nuevas aportaciones al trabajo presentado:

- *Tamaño de páginas del firmware*. Los paquetes de datos en los que se envían las páginas del *firmware* pueden ser configurados para que tengan los siguientes tamaños: 32, 64, 128 y 192 bytes. En este trabajo de investigación, al igual que en las pruebas de campo realizadas para la validación del modelo de simulación, la imagen del *firmware* se segmenta en páginas de 64 bytes. Como trabajo futuro, se estudiará la posibilidad de realizar experimentos empleando otros tamaños de página (32, 128 y 192 bytes) y analizar de qué forma afecta el tamaño de las mismas al proceso de actualización *firmware* (como por ejemplo, reduciendo/aumentando la duración del proceso de actualización, el número de colisiones, la disponibilidad de la subred etc.).

- *Espaciado temporal del envío de páginas*. La separación temporal con la que el nodo base transmite las páginas del *firmware* es de libre configuración. En las simulaciones realizadas, al igual que en los experimentos sobre redes reales llevados a cabo para la validación del modelo de simulación, se ha fijado una separación temporal de 600 ms. Como trabajo futuro, se podrán realizar experimentos variando este valor para poder analizar de qué forma afecta el espaciado temporal de las páginas al proceso de actualización *firmware* (como por ejemplo reduciendo o aumentando el número de colisiones entre paquetes).

- *Número de páginas del firmware consecutivas a enviar*. Cuando el nodo base le solicita a un nodo de servicio que le envíe la lista de páginas del *firmware* que le quedan por recibir y el nodo base recibe la respuesta, comienza con el envío de las páginas indicadas. El número máximo de páginas a enviar por el nodo base antes de volver a escoger otro nodo de servicio y solicitarle una nueva lista de páginas por recibir es libre, y en el trabajo de investigación presentado el valor empleado es de 512. Como trabajo futuro se podrá estudiar la influencia que tiene el valor de este parámetro en el proceso de actualización *firmware* mediante la realización de experimentos empleando distintos valores del mismo (como por ejemplo, en el número de retransmisiones de páginas del *firmware*, reducción/aumento de la duración del proceso de actualización, etc.).

- *Algoritmo de selección del nodo servicio al que solicitar la lista páginas por recibir*. Cuando comienza la fase de descarga del *firmware* el nodo base escoge un nodo de servicio al que le solicita la lista de páginas del *firmware* que le quedan por recibir. En el trabajo de investigación realizado, la selección de este nodo se realiza de forma aleatoria entre aquellos nodos que aún no han sido actualizados y que hayan sido inicializados en la presente ronda de actualización. Como trabajo futuro, se estudiará la posibilidad de implementar un algoritmo de selección de nodos de servicio

más avanzado, que por ejemplo priorice la selección de nodos de niveles más alejados del nodo base, de modo que pueda analizarse la influencia de este algoritmo en el comportamiento del proceso de actualización *firmware*.

- *Implementación y evaluación de las estrategias en despliegues reales*. En el capítulo 5, las estrategias descritas se han simulado sobre escenarios con distinta topología física y lógica. Como trabajo futuro, se puede estudiar la posibilidad de escoger distintos escenarios reales con topologías similares e implementar sobre equipos reales las estrategias propuestas en este trabajo de investigación así como otras que puedan surgir para ver si las conclusiones obtenidas de las simulaciones se asemejan a las obtenidas en pruebas de campo.

Apéndice

A

Información adicional del estándar PRIME

Este apéndice recoge información adicional relativa al estándar PRIME que no ha sido incluida en el capítulo 2 y se estructura como sigue. La sección A.1 se centra en los aspectos que tienen que ver con la capa PHY, la sección A.2 añade detalles de la capa MAC, y por último, la sección A.3 incluye información adicional del nivel de gestión.

A.1 Capa PHY

A.1.1 Trama PHY

La capa física (PHY) recibe una trama MPDU de la capa MAC y genera una trama PHY (preámbulo + trama PPDU) cuya estructura se muestra en la figura A.1.

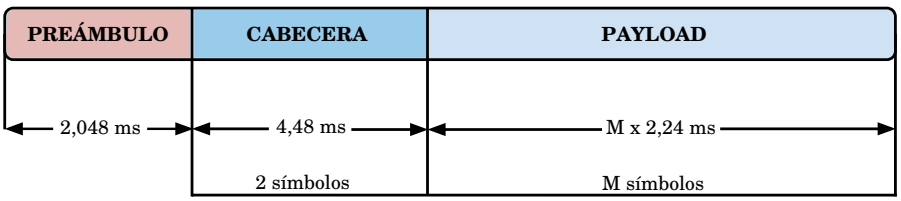


Figura A.1: Estructura de la trama PHY

Cada trama PHY comienza con un preámbulo de 2,048 ms de duración, seguido por un determinado número de símbolos OFDM, cada uno de los cuales

dura 2,24 ms. Los primeros dos símbolos OFDM corresponden a la cabecera de la trama PHY, mientras que los M símbolos OFDM restantes transportan la carga útil o *payload*. El valor de M está indicado en uno de los campos de la cabecera y tiene un valor máximo de 63.

La cabecera se compone de dos símbolos OFDM que siempre se envían utilizando la modulación DBPSK con el FEC (código convolucional) activado. Sin embargo, para el envío del *payload* se pueden emplear las modulaciones DBPSK, DQPSK o DQPSK dependiendo de la opción escogida por la capa MAC. La capa MAC escogerá el mejor esquema de modulación en función de los errores detectados en transmisiones anteriores realizadas, o utilizando el SNR. Esto incluye decidir si utilizar o no el mecanismo FEC.

La figura A.2 muestra la estructura de la cabecera PHY y el *payload* de la trama PPDU.

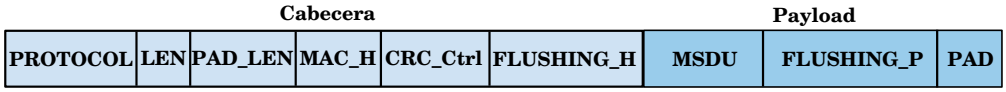


Figura A.2: Cabecera y *payload* de la trama PPDU

- **Cabecera PHY**.
  - **PROTOCOL**. Indica el esquema de transmisión empleado por el *payload*.
  - **LEN**. Indica la longitud del *payload* en símbolos OFDM. Rellenado por la capa PHY.
  - **PAD_LEN**. Indica la longitud del campo *PAD* en bytes. Rellenado por la capa PHY.
  - **MAC_H**. Cabecera de la capa MAC. Se incluye en los símbolos de la cabecera para proteger la información que contiene.
  - **CRC_Ctrl**. CRC de los campos PROTOCOL, LEN, PAD_LEN y MAC_H.
  - **FLUSHING_H**. Bits de *flushing* que se necesitan para la para la decodificación convolucional. Todos los bits de este campo están a cero con el fin de reiniciar el codificador.
- ***Payload***.
  - **MSDU**. *MAC Service Data Unit* sin codificar.
  - **Flushing_P**. Este campo sólo está presente cuando el FEC está activado. Todos los bits de este campo están a cero.
  - **PAD**. Campo de *padding*. Para asegurar que el número de símbolos OFDM generados es un número entero se añaden estos bits de relleno de valor cero.

A.2 Capa MAC

A.2.1 Trama MAC PDU genérica (GPDU)

La trama MAC PDU genérica (GPDU) consta de una cabecera MAC genérica, seguido de uno o más paquetes MAC y un CRC al final (figura A.3). En los siguientes subapartados se añadirá información adicional relativa a la estructura de la cabecera MAC genérica y paquetes MAC que servirán para completar la información incluida en el capítulo 2.

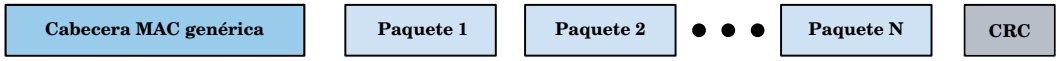


Figura A.3: Estructura de la trama GPDU

A.2.1.1 Cabecera MAC genérica

La trama GPDU consta de una cabecera MAC genérica cuya estructura se muestra en la figura A.4. Por otro lado, la tabla A.1 describe los campos componen dicha cabecera.

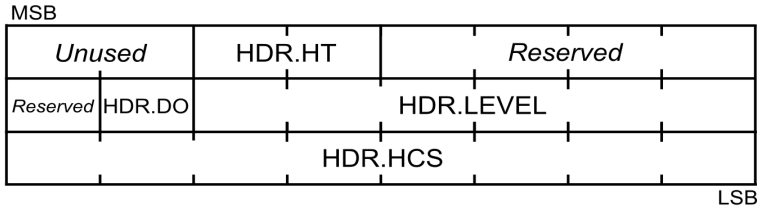


Figura A.4: Estructura de la cabecera MAC genérica

A.2.1.2 Paquete

Los paquetes de una trama GPDU tienen una estructura que se compone de una cabecera de 6 bytes y un *payload*. La cabecera de un paquete tiene la estructura que se muestra en la figura A.5 y la tabla A.2 muestra los campos de dicha cabecera.

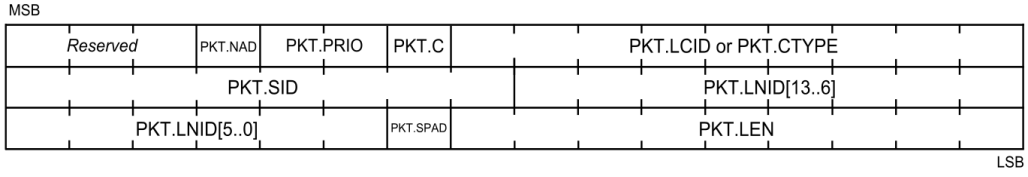


Figura A.5: Estructura de la cabecera de paquete

Tabla A.1: Campos de la cabecera MAC genérica

Nombre	**Longitud**	**Descripción**
Unused	2 bits	Estos bits no se usan por lo que están siempre a 0; se incluyen para realizar el alineamiento con el campo MAC_H en la cabecera PPDU.
HDR.HT	2 bits	Tipo de cabecera. HDR.HT = 0 para tipo GPDU
Reserved	5 bits	Siempre a 0 para esta versión de la especificación. Reservado para uso futuro.
HDR.DO	1 bit	HDR.DO = 1 si el MAC PDU es de tipo *downlink* HDR.DO = 0 si si el MAC PDU es de tipo *uplink*.
HDR.LEVEL	6 bits	Nivel del PDU en la jerarquía de conmutación. Los paquetes entre el nivel 0 y el nodo base son de HDR.LEVEL = 0. Los paquetes entre los niveles k y k-1 son de HDR.LEVEL = k. Además si HDR.DO = 0, HDR.LEVEL representa el nivel del transmisor del paquete. Si HDR.DO = 1, HDR.LEVEL representa el nivel del receptor del paquete.
HDR.HCS	8 bits	*Header Check Sequence*. Se trata de un campo para detectar errores en la cabecera y chequear que el MAC PDU es de esta subred.

Tabla A.2: Campos de la cabecera de paquetes

Nombre	**Longitud**	**Descripción**
Reserved	3 bits	Siempre 0 para esta versión de la especificación. Reservado para uso futuro.
PKT.NAD	1 bit	Agregación de paquetes Si PKT.NAD = 0 el paquete puede ser agregado con otros paquetes en destino. Si PKT.NAD = 1 el paquete no debe ser agregado con otros paquetes en destino.
PKT.PRIO	2 bits	Indica la prioridad del paquete entre 0 y 3.
PKT.C	1 bit	Control Si PKT.C = 0 es un paquete de datos Si PKT.C = 1 es un paquete de control
PKT.LCID/PKT.CTYPE	9 bits	*Local Connection Identifier* o *Control Type*.

		Si PKT.C = 0, PKT.LCID representa el Local Connection Identifier del paquete de datos. Si PKT.C = 1, PKT.CTYPE representa el tipo de paquete de control
PKT.SID	8 bits	Identificador de *Switch*. Si HDR.DO = 0 , PKT.SID representa el SID de la fuente del paquete. Si HDR.DO = 1, PKT.SID representa el SID del destino del paquete.
PKT.LNID	14 bits	*Local Node Identifier*. Si HDR.DO = 0, PKT.LNID representa el LNID de la fuente del paquete. Si HDR.DO = 1, PKT.LNID representa el LNID del destino del paquete.
PKT.SPAD	1 bit	Indica si se han insertado bits de relleno a la hora de encriptar el *payload*. Este bit solo es importante para el perfil de seguridad.
PKT.LEN	9 bits	Longitud del *payload* del paquete en bytes.

A.2.2 PNPDU

Además de las tramas GPDU existen otro tipo de tramas MAC entre las que se encuentran las tramas PNPDU. La figura A.6 muestra la estructura de este tipo de tramas. Por otro lado, la tabla A.3 incluye los campos que componen dicha trama.

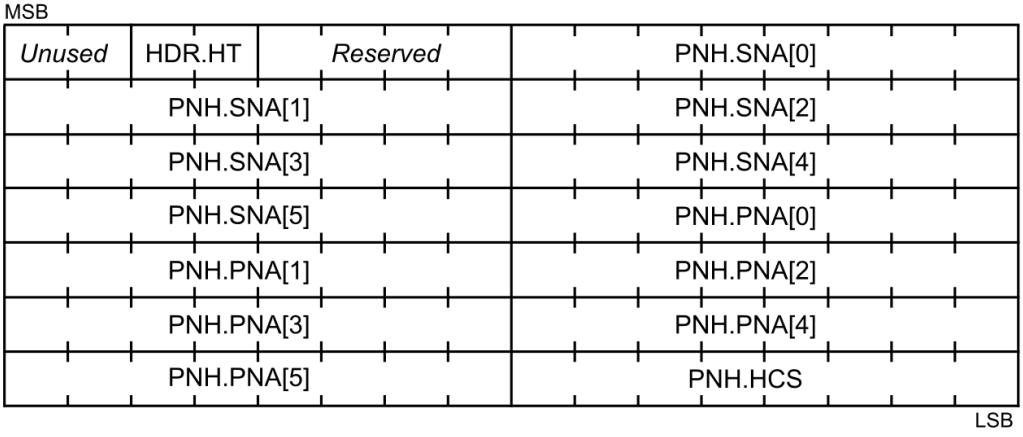


Figura A.6: Estructura de la trama PNPDU

Tabla A.3: Campos de la trama PNPDU

Nombre	Longitud	Descripción
Unused	2 bits	Bits sin utilizar que se ponen a 0. Se incluyen para llevar a cabo el alineamiento con el campo MAC_H de la cabecera PPDU.
HDR.HT	2 bits	*Header type*. HDR.HT = 1 para Promotion Needed MAC PDU
Reserved	4 bits	Siempre cero para esta versión de la especificación. Reservado para uso futuro.
PNH.SNA	48 bits	Dirección de subred. El EU-48 del nodo base de la subred en la que se encuentra el nodo de servicio que está intentando conectarse. FF:FF:FF:FF:FF:FF para solicitar la promoción en una subred disponible. SNA[0] es el byte más significativo y SN[5] el menor.
PNH.PNA	48 bits	*Promotion need address*. El EUI-48 del nodo que necesita la promoción. Es el EUI-48 del emisor del mensaje.
PNH.HCS	8 bits	*Header Check Sequence*. Se trata de un campo para detectar errores en la cabecera. El transmisor calcula el *PNH.HCS* de los primeros 13 bytes de la cabecera e inserta el resultado en el campo PNH.HCS (último byte de la cabecera).

A.2.3 BPDU

Las tramas BPDU o balizas tienen la estructura que se muestra en la figura A.7. Por otro lado, los campos de la trama se incluyen en la tabla A.4.

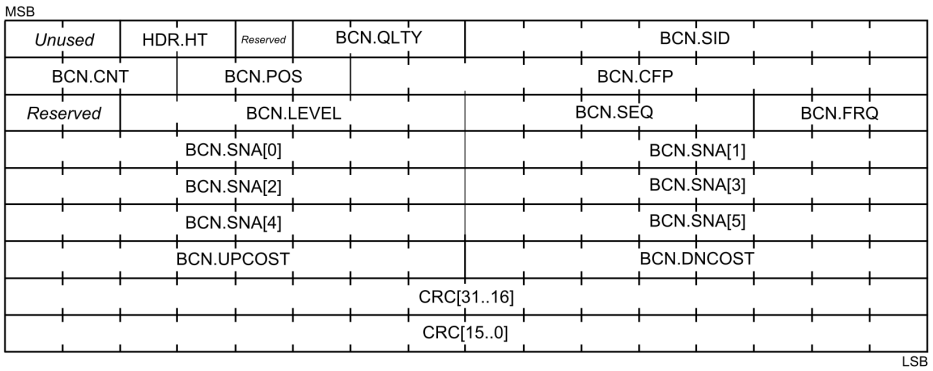


Figura A.7: Estructura de la trama Beacon PDU

Tabla A.4: Campos de la trama BPDU

Nombre	**Longitud**	**Descripción**
Unused	2 bits	Bits sin utilizar, se ponen a 0; se incluyen para llevar a cabo el alineamiento con el campo MAC_H en la cabecera PPDU.
HDR.HT	2 bits	Header type HDR.HT = 2 para Beacon PDU
Reserved	1 bit	Siempre cero para esta versión de la especificación. Reservado para uso futuro.
BCN.QLTY	3 bits	Calidad de la conectividad *round-trip* desde este *Switch* al nodo base. BCN.QLTY = 7 para la calidad más alta (es el nodo base o un muy buen nodo *Switch*), BCN.QLTY = 0 para la calidad más baja (*Switch* que tiene una conexión inestable).
BCN.SID	8 bits	Identificador del *Switch* emisor.
BCN.CNT	8 bits	Número de *beacon-slots* en esta trama.
BCN.SLT	3 bits	*Beacon-slot* en el que se ha transmitido este BPDU. BCN.SLT = 0 está reservado para el nodo base.
BCN.CFP	10 bits	*Offset* del CFP desde el comienzo de la trama. BCN.CFP = 0 indica la ausencia de CFP en la trama.
Reserved	1 bit	Siempre cero para esta versión de la especificación. Reservado para uso futuro.
BCN.LEVEL	6 bits	Nivel en la jerarquía del nodo *Switch* que transmite el BPDU.
BCN.SEQ	5 bits	Número de secuencia de este BPDU en la super trama. Incrementado por cada *beacon* que envía el nodo base y se propaga por los *Switches* a través de su BPDU de forma que la subred entera tiene la noción del número de secuencia en cada momento.
BCN.FRQ	3 bits	Frecuencia de transmisión de este BPDU. 0 = 1 *beacon* cada trama 1 = 1 *beacon* cada 2 tramas 2 = 1 *beacon* cada 4 tramas 3 = 1 *beacon* cada 8 tramas 4 = 1 *beacon* cada 16 tramas 4 = 1 *beacon* cada 16 tramas 5 = 1 *beacon* cada 32 tramas 6 = Reservado

		7 = Reservado
BCN.SNA	48 bits	Identificador de la subred en la que el *Switch* está transmitiendo el BPDU.
BCN.UPCOST	8 bits	El coste total de *uplink* desde el nodo *Switch* transmisor al nodo base. El coste de un simple salto se calcula en base al esquema de modulación empleado en ese salto en dirección *uplink* (de subida). 8PSK = 0 QPSK = 1 BPSK = 2 8PSK_F =1 QPSK_F = 2 BPSK_F = 4 El nodo base transmitirá su *beacon* con un coste de 0 (BCN.UPCOST = 0). Un nodo *Switch* transmitirá en su *beacon* el valor recibido en su nodo *Switch* superior más el coste que supone llegar hasta ese *Switch*. Cuando este valor supera el máximo simplemente se pone el máximo valor que admite este campo.
BCN.DNCOST	8 bits	El coste total de *downlink* desde el nodo base transmisor al nodo *Switch*. El coste de un simple salto se calcula en base al esquema de modulación empleado en ese salto en dirección *downlink* (de bajada) 8PSK = 0 QPSK = 1 BPSK = 2 8PSK_F =1 QPSK_F = 2 BPSK_F = 4 El nodo base transmitirá su *beacon* con un coste de 0 (BCN.DNCOST = 0). Un nodo *Switch* transmitirá en su *beacon* el valor recibido en su nodo *Switch* superior más el coste que supone llegar hasta ese *Switch*. Cuando este valor supera el máximo simplemente se pone el máximo valor que admite este campo.

CRC	32 bits	El CRC se calcula con el mismo algoritmo definido para el campo CRC de la trama MAC PDU. Este CRC se calcula sobre el BPDU sin el campo CRC.

A.2.4 Constantes de la capa MAC

En la tabla A.5 se incluyen algunas de las constantes de la capa MAC que se han ido citando a lo largo del capítulo 2.

Tabla A.5: Constantes de la capa MAC

Constante	**Valor**	**Descripción**
MACBeaconLength	4 símbolos	Longitud de cada baliza.
MACMinSCPLength	64 símbolos	Longitud mínima del periodo SCP.
MACFrameLength	276 símbolos	Longitud de una trama MAC.
MACPriorityLevels	4	Número de prioridades soportados por el sistema.
MACMaxPRNIgnore	3	Número de paquetes *PNPDU* que puede ignorar un nodo de servicio.
$N_{miss-beacon}$	5	Número de veces que un nodo de servicio no recibe una baliza esperada antes de considerar a su nodo *Switch* como no disponible.

A.3 Nivel de gestión

En este apartado, se incluye alguna información adicional del nivel de gestión que completa lo descrito en el capítulo 2.

A.3.1 Atributos PIB de la capa MAC

A continuación, la tabla A.6 muestra algunos de los atributos PIB de la capa MAC mencionados a lo largo de este trabajo de investigación.

Tabla A.6: Atributos PIB de la capa MAC

Constante	**Rango de valores**	**Descripción**	**Valor por defecto**
macSCPMaxTxAttempts	2-5	Número de veces que se ejecuta el algoritmo CSMA/CA antes de determinar que el canal está ocupado.	5
macMaxPromotionPdu	1-4	Número máximo de PNPDUs que puede enviar un nodo de servicio durante *macPromotionPduTxPeriod* segundos.	2
macPromotionPduTxPeriod	2-8 segundos	Cantidad de tiempo que limita el envío de PNPDUs de un nodo de servicio.	5
macMinSwitchSearchTime	16-32 segundos	Mínimo tiempo en el que el nodo de servicio en el estado *Desconectado* escanea la red en busca escuchar balizas antes de transmitir una trama *PNPDU*.	24
macCtlReTxTimer	2-20 segundos	Número de segundos que una entidad MAC espera el acuse de recibo del paquete de control MAC de su entidad par. Al término de este tiempo, la entidad MAC puede retransmitir el paquete de control MAC.	15

macMaxCtlReTx	3-5	El número máximo de veces que una entidad MAC intentará retransmitir un paquete de control del que no ha recibido acuse de recibo. Si el recuento de retransmisiones alcanza este máximo, la entidad MAC deberá abortar cualquier nuevo intento de transmitir el paquete de control MAC.	3

A.3.2 Estructura de paquetes de control del proceso de actualización *firmware*

En el presente apartado se incluye la estructura de paquetes de control que intervienen en el proceso de actualización *firmware* que describe el estándar PRIME.

A.3.2.1 FU_INIT_REQ

La tabla A.7 muestra la estructura del paquete de inicialización (*FU_INIT_REQ*) del proceso de actualización *firmware*.

Tabla A.7: Campos del paquete *FU_INIT_REQ*

Campo	**Longitud**	**Descripción**
Type	4 bits	0 = FU_INIT_REQ
Versión	2 bits	0 para esta versión del protocolo.
PageSize	32 bits	0 para PageSize = 32 1 para PageSize = 64 2 para PageSize = 128 3 para PageSize = 192
ImageSize	32 bits	Tamaño de la imagen en bytes.
CRC	32 bits	CRC de la imagen del firmware.

A.3.2.2 FU_EXEC_REQ

La tabla A.8 muestra la estructura del paquete *FU_EXEC_REQ* empleado para indicar al nodo de servicio el momento en el que tiene que reiniciarse y comenzar a utilizar el nuevo *firmware*.

Tabla A.8: Campos del paquete *FU_EXEC_REQ*

Campo	Longitud	Descripción
Type	4 bits	1 = FU_EXEC_REQ
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
RestartTimer	16 bits	0..65536 segundos. Tiempo antes del reinicio con el nuevo *firmware*.
SafetyTimer	16 bits	0..65536 segundos. Tiempo de prueba del nuevo *firmware*. La cuenta comienza cuando el nodo de servicio pasa al estado “Upgrade”.

A.3.2.3 FU_CONFIRM_REQ

La tabla A.9 muestra la estructura del paquete de confirmación de la nueva versión del *firmware* o paquete *FU_CONFIRM_REQ*.

Tabla A.9: Campos del paquete *FU_CONFIRM_REQ*

Campo	Longitud	Descripción
Type	4 bits	2 = FU_CONFIRM_REQ
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0

A.3.2.4 FU_STATE_REQ

La tabla A.10 muestra la estructura del paquete *FU_STATE_REQ* que envía el nodo base para preguntar a un nodo de servicio acerca de su estado en el proceso de actualización.

Tabla A.10: Campos del paquete *FU_STATE_REQ*

Campo	Longitud	Descripción
Type	4 bits	3 = FU_STATE_REQ
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0

A.3.2.5 FU_KILL_REQ

La tabla A.11 muestra la estructura del paquete *FU_KILL_REQ* que envía el nodo base cuando quiere finalizar el proceso de actualización *firmware*.

Tabla A.11: Campos del paquete *FU_KILL_REQ*

Campo	Longitud	Descripción
Type	4 bits	4 = FU_KILL_REQ
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0

A.3.2.6 FU_STATE_RSP

La tabla A.12 muestra la estructura del paquete *FU_STATE_RSP* enviado por los nodos de servicio.

A.3.2.7 FU_DATA

La tabla A.13 muestra la estructura del paquete enviado por el nodo base a los nodos de servicio para la transmisión de cada página del *firmware*.

A.3.2.8 FU_MISS_REQ

La tabla A.14 muestra la estructura del paquete *FU_MISS_REQ* enviado por el nodo base a los nodos de servicio.

A.3.2.9 FU_MISS_BITMAP

La tabla A.15 muestra la estructura del paquete *FU_MISS_BITMAP* enviado por un nodo de servicio en respuesta a un paquete *FU_MISS_REQ*. Este paquete incluye un *bitmap* que contiene información acerca del estado de cada página del *firmware*.

Tabla A.12: Campos del paquete *FU_STATE_RSP*

Campo	Longitud	Descripción
Type	4 bits	5 = FU_STATE_RSP
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
State	4 bits	0 para el “Idle” 1 para “Receiving” 2 para “Complete” 3 para “Countdown” 4 para “Upgrade” 5 reservado para uso futuro
Reserved	4 bits	0
CRC	32 bits	CRC (el mismo que el del paquete *FU_INIT_REQ*).
Received	32 bits	Número de páginas recibidas (este campo solo está presente si el estado del nodo es “Receiving").

Tabla A.13: Campos del paquete *FU_DATA*

Campo	Longitud	Descripción
Type	4 bits	6 = FU_DATA
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
PageIndex	32 bits	Índice de la página a transmitir.
Reserved	8 bits	Bits de relleno. Su valor por defecto es 0.
Data	Variable	Los datos correspondientes a la página. La longitud de este campo es *PageSize* (32, 64, 128 o 192) bytes para todas las páginas, excepto para la última página que tendrá el resto de los bytes.

A.3.2.10 FU_MISS_LIST

La tabla A.16 muestra la estructura del paquete *FU_MISS_LIST* enviado por un nodo de servicio en respuesta a un paquete *FU_MISS_REQ*. Este paquete incluye lista de páginas del *firmware* que le quedan por recibir al nodo de servicio.

Tabla A.14: Campos del paquete *FU_MISS_REQ*

Campo	Longitud	Descripción
Type	4 bits	7 = FU_MISS_REQ
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
PageIndex	32 bits	Punto de inicio desde donde se recoge la información de las páginas que faltan.

Tabla A.15: Campos del paquete *FU_MISS_BITMAP*

Campo	Longitud	Descripción
Type	4 bits	8 = FU_MISS_BITMAP
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
PageIndex	32 bits	Índice de página de la página representada por el primer bit del mapa de bits. Debe ser el mismo que el campo *PageIndex* en los mensajes FU_MISS_REQ , o posterior. Si es posterior, significa que las páginas que están en medio han sido recibidas correctamente. En este caso, si todas las páginas que están después del *PageIndex* indicado o por el indicado en el paquete *FU_MISS_REQ* se han recibido correctamente, el nodo de servicio comenzará a mirar desde el principio (*PageIndex* = 0).
Bitmap	Variable	Este *bitmap* contiene la información acerca del estado de cada página. El primer bit (bit más significativo del primer byte) representa el estado de *PageIndex* + 1 y de ahí en adelante. El valor 1 representa que falta una página mientras que el valor 0 indica que la página ya ha sido recibida. El tamaño máximo de este campo es *PageSize* bytes.

A.3.2.11 FU_INFO_REQ

La tabla A.17 muestra la estructura del paquete de solicitud de información (*FU_INFO_REQ*) enviado por el nodo base a un nodo de servicio.

A.3.2.12 FU_INFO_RSP

Las tablas A.18 muestran la estructura del paquete de respuesta *FU_INFO_RSP* enviado por un nodo de servicio en respuesta a la solicitud del información enviada por el nodo base. Por otro lado, la tabla A.19 incluye los campos de entra-

Tabla A.16: Campos del paquete *FU_MISS_LIST*

Campo	Longitud	Descripción
Type	4 bits	9 = FU_MISS_LIST
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
PageIndexList	Variable	Lista de páginas que quedan aún por recibir. Cada página está representada por su *PageIndex*, codificado como un entero de 32 bits. Estas páginas deberán estar ordenadas en orden ascendente. El primer índice de página deberá ser igual que el campo *PageIndex* del paquete *FU_MISS_REQ* o posterior. Si es posterior significa que las páginas que están en medio ya han sido recibidas. El tamaño máximo de este campo es *PageSize* bytes.

Tabla A.17: Campos del paquete *FU_INFO_REQ*

Campo	Longitud	Descripción
Type	4 bits	10 = FU_INFO_REQ
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
InfoIdList	Variable	Lista de identificadores con la información que se quiere extraer. Cada identificador tiene 1 byte de longitud. La longitud máxima de este campo es 32 bytes.

da del campo *InfoData* del paquete *FU_INFO_RSP*, mientras que la tabla A.20 incluye los valores posibles para *InfoId* del campo *InfoData*.

A.3.2.13 FU_CRC_REQ

La tabla A.21 muestra la estructura del paquete de solicitud de comprobación de la integridad de la imagen del *firmware* (*FU_CRC_REQ*) enviado por el nodo base.

A.3.2.14 FU_CRC_RSP

La tabla A.22 muestra la estructura del paquete de respuesta *FU_CRC_RSP* a la solicitud de comprobación de la imagen del *firmware* enviado por el nodo base.

Tabla A.18: Campos del paquete *FU_INFO_RSP*

Campo	**Longitud**	**Descripción**
Type	4 bits	11 = FU_INFO_RSP
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
InfoData	0 - 192 bytes	Datos que contienen la información solicitada por el nodo base. Puede contener varias entradas (uno por cada identificador solicitado) teniendo cada una de ellas un tamaño máximo de 32 bytes. El tamaño máximo de este campo es de 192 bytes (6 entradas)

Tabla A.19: Campos de cada entrada *InfoData* del paquete *FU_INFO_RSP*

Campo	**Longitud**	**Descripción**
InfoId	8 bits	Identificador de la información incluida en la tabla A.20.
Reserved	3 bits	0
Length	5 bits	Longitud del campo *Data*.
Data	0 - 30 bytes	Información proporcionada por el nodo de servicio. El contenido de este campo depende del significado del campo *InfoId*, pero en ningún caso deberá superar los 30 bytes.

Tabla A.20: Valores posibles para InfoId

InfoId	**Nombre**	**Descripción**
0	Fabricante	Identificador universal del fabricante.
1	Modelo	Modelo del producto que trabaja como nodo de servicio.
2	Firmware	Versión actual del *firmware*.
128-255	Específico del fabricante	Rango de valores que son propios del fabricante.

Tabla A.21: Campos del paquete *FU_CRC_REQ*

Campo	**Longitud**	**Descripción**
Type	4 bits	12 = FU_CRC_REQ
Versión	2 bits	0 para esta versión del protocolo.
Reserved	2 bits	0
SectionSize	32 bits	Tamaño de la imagen del *firmware* en bytes.
CRC	32 bits	CRC de la imagen del *firmware*.

Tabla A.22: Campos del paquete *FU_CRC_RSP*

Campo	**Longitud**	**Descripción**
Type	4 bits	13 = FU_CRC_RSP
Versión	2 bits	0 para esta versión del protocolo.
CRC_Result	1 bits	Resultado del CRC “0” Error “1” Correcto
Reserved	1 bit	0

Bibliografía

[ABS+10] Aitor Arzuaga, Iñigo Berganza, Alberto Sendín, Manu Sharma, and Badri Varadarajan. PRIME interoperability tests and results from field. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 126–130. IEEE, 2010.

[AKYN05] Kaywan H. Afkhamie, Srinivas Katar, Larry Yonge, and Richard Newman. An overview of the upcoming HomePlug AV standard. In *Power Line Communications and Its Applications, 2005 International Symposium on*, pages 400–404. IEEE, 2005.

[All01] Home Plug Alliance. Homeplug 1.0 Technology Whitepaper, 2001.

[AMRMA14] Eduardo Alonso, Javier Matanza, Carlos Rodríguez-Morcillo, and Sadot Alexandres. A switch promotion algorithm for improving PRIME PLC network latency. In *Power Line Communications and its Applications (ISPLC), 2014 18th IEEE International Symposium on*, pages 278–283. IEEE, 2014.

[AMRMA15] Eduardo Alonso, Javier Matanza, Carlos Rodríguez-Morcillo, and Sadot Alexandres. Performance evaluation of AMR simultaneous polling strategies in a PRIME PLC network. In *Power Line Communications and Its Applications (ISPLC), 2015 IEEE International Symposium on*, pages 101 – 106. IEEE, 2015.

[Ben03] Dirk Benyoucef. A new statistical model of the noise power density spectrum for powerline communication. In *Proceedings of the 7th International Symposium on Power-Line Communications and its Applications, Kyoto, Japan*, pages 136–141, 2003.

[Ber08] Iñigo Berganza. Iberdrola Strategy on AMI and Smart Grids. *Metering International*, (2), 2008.

[C+80] Power System Communications Committee et al. Summary of an IEEE guide for power-line carrier applications. *IEEE Trans. on Power Apparatus and Systems*, pages 2334–7, 1980.

[CDCL09] José Antonio Cortés, Luis Díez, Francisco Javier Cañete, and Jesús López. Analysis of the periodic impulsive noise asynchronous with the mains in indoor PLC channels. In *Power Line Communications and Its Applications, 2009. ISPLC 2009. IEEE International Symposium on*, pages 26–30. IEEE, 2009.

[CEN93] EN CENELEC. 50065-1:"signalling on low voltage electrical installations in the frequency range 3 khz to 148.5 khz". *Part 1: General requirements, frequency bands and electromagnetic disturbances, European Committee for Electrotechnical Standardization*, 1993.

[DARM11] Javier Matanza Domingo, Sadot Alexandres, and Carlos Rodríguez-Morcillo. PRIME performance in power line communication channel. In *Power Line Communications and Its Applications (ISPLC), 2011 IEEE International Symposium on*, pages 159–164. IEEE, 2011.

[DBDAT14] Luca Di Bert, Salvatore D Alessandro, and Andrea M Tonello. A G3-PLC simulator for access networks. In *Power Line Communications and its Applications (ISPLC), 2014 18th IEEE International Symposium on*, pages 99–104. IEEE, 2014.

[DBS11] Dacfey Dzung, Iñigo Berganza, and Alberto Sendín. Evolution of powerline communications for smart distribution: from ripple control to OFDM. In *Power Line Communications and Its Applications (ISPLC), 2011 IEEE International Symposium on*, pages 474–478. IEEE, 2011.

[Dos01] Klaus Dostert. *Powerline communications*. Prentice Hall PTR, 2001.

[end44] *Endesa*, 1944. `https://www.endesaclientes.com/hogares.html` (Último acceso 24 de septiembre 2015).

[Eur10] European Commission. 2010. Europe 2020. A strategy for smart, sustainable and inclusive growth. Brussels: European Commission, 2010.

[Eyr90] BE Eyre. Results of a comprehensive field trial of a United Kingdom customer telemetry system using mains borne signaling.

In *Proceedings of the Sixth International Conference on Metering Apparatus and Tariffs for Electricity Supply, IEE Conference Publication*, number 317, pages 252–256, 1990.

[FLNS10] Hendrik C. Ferreira, Lutz Lampe, John Newbury, and Theo G. Swart. *Power Line Communications: Theory and Applications for Narrowband and Broadband Communications Over Power Lines*. Wiley, 2010.

[FOSEUG+13] A. Fernández Olivera, A. Sendín Escalona, I. Urrutia Galdós, J. Mateo Arenas, P. Angueira Buceta, and JJ. Ferro Vázquez. Analysis of PRIME PLC smart metering networks performance. *Renewable Energy and Power Quality Journal*, (11), 2013.

[Fox15] John Fox. *Applied regression analysis and generalized linear models*. Sage Publications, 2015.

[G+10] IEEE 1901 Working Group et al. IEEE Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications. Technical report, Tech. Report, 2010.

[g3p09] *G3-PLC*, 2009. `http://g3-plc.com/` (Úlimo acceso 20 diciembre 2014).

[G.909] ITU-T Rec. G.9960. Unified high-speed wire-line based home networking transceivers-foundation, 2009.

[G.910] ITU-T Rec. G.9961. Data link layer (dll) for unified high-speed wire-line based home networking transceiver, 2010.

[gad09] *Gestión Activa de la Demanda*, 2009. `http://gad.ite.es/index_en.html` (Último acceso 29 diciembre 2014).

[GFB+04] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users'Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[GKK08] Stefano Galli, Hisao Koga, and Nobutaka Kodama. Advanced signal processing for plcs: Wavelet-ofdm. In *Power Line Communications and Its Applications, 2008. ISPLC 2008. IEEE International Symposium on*, pages 187–192. IEEE, 2008.

[GL08] Stefano Galli and Oleg Logvinov. Recent developments in the standardization of power line communications within the IEEE. *Communications Magazine, IEEE*, 46(7):64–71, 2008.

[GLT99] William Gropp, Ewing Lusk, and Rajeev Thakur. *Using MPI-2: Advanced features of the message-passing interface*. MIT press, 1999.

[GMB+14] Asmir Gogic, Amar Mahmutbegovic, Dzemo Borovina, Ismail Hakki Cavdar, and Nermin Suljanovic. Simulation of the narrow-band PLC system implementing PRIME standard. In *Energy Conference (ENERGYCON), 2014 IEEE International*, pages 1520–1525. IEEE, 2014.

[GRD04] Matthias Gotz, Manuel Rapp, and Klaus Dostert. Power line channel characteristics and their effect on communication system design. *Communications Magazine, IEEE*, 42(4):78–86, 2004.

[GSW11] Stefano Galli, Anna Scaglione, and Zhifang Wang. For the grid and through the grid: The role of power line communications in the smart grid. *Proceedings of the IEEE*, 99(6):998–1027, 2011.

[GT10] Shmuel Goldfisher and Shinji Tanabe. IEEE 1901 access system: An overview of its uniqueness and motivation. *Communications Magazine, IEEE*, 48(10):150–157, 2010.

[GYC+08] Q. Gao, J.Y. Yu, P.H.J. Chong, P.L. So, and E. Gunawan. Solutions for the "silent node" problem in an automatic meter reading system using power-line communications. *Power Delivery, IEEE Transactions on*, 23(1):150–156, 2008.

[HC90] S.J. Holmes and D. Campbell. Communicating with domestic electricity meters. In *Metering Apparatus and Tariffs for Electricity Supply, 1990., Sixth International Conference on*, pages 129–133. IET, 1990.

[HHL05] Halid Hrasnica, Abdelfatteh Haidine, and Ralf Lehnert. *Broadband powerline communications: network design*. John Wiley & Sons, 2005.

[Hoo98] Olaf G. Hooijen. On the relation between network-topology and power line signal attenuation. In *International Symposium on Power Line Communications*, pages 45–56, 1998.

[HS00] Christian Hensen and Wolfgang Schulz. Time dependence of the channel characteristics of low voltage power-lines and its effects on hardware implementation. *AEU-ARCH. ELEKTR. UBERTRAGUNGSTECHN*, 54(1):23–32, 2000.

[KH+10] Kim Kyong-Hoe et al. PHY abstraction methodology for the performance evaluation of PLC channels. In *IEEE International Symposium on Power Line Communications and Its Applications (ISPLC 2010). Rio de Janeiro*, pages 28–31, 2010.

[KHM11] Mehdi Korki, Nasser Hosseinzadeh, and Taleb Moazzeni. Performance evaluation of a narrowband power line communication for smart grid with noise reduction technique. *Consumer Electronics, IEEE Transactions on*, 57(4):1598–1606, 2011.

[kon02] Konnex (knx) standard: Konnex assoc, 2002.

[LnLUS15] Janire Larrañaga, Jon Legarda, Iker Urrutia, and Alberto Sendín. An experimentally validated PRIME subnetwork simulation model for utility applications. In *Power Line Communications and Its Applications (ISPLC), 2015 IEEE International Symposium on*, pages 95–100. IEEE, 2015.

[M+10] Anil Mengi et al. Successive impulsive noise suppression in OFDM. In *Power Line Communications and Its Applications (ISPLC), 2010 IEEE International Symposium on*, pages 33–37. IEEE, 2010.

[Mak01] Sioe T. Mak. Power delivery infrastructure differences and their impact on different types of power line communications for automatic meter reading. In *Electricity Distribution, 2001. Part 1: Contributions. CIRED. 16th International Conference and Exhibition on (IEE Conf. Publ No. 482)*, volume 4, pages 5–pp. IET, 2001.

[MARM13] Javier Matanza, Sadot Alexandres, and Carlos Rodríguez-Morcillo. Automatic meter-reading simulation through power line communication. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, pages 283–287. IEEE, 2013.

[met10] *Meters and More (M&M)*, 2010. `http://www.metersandmore.com/` (Último acceso 20 diciembre 2014).

[Mis86] Jayadev Misra. Distributed discrete-event simulation. *ACM Computing Surveys (CSUR)*, 18(1):39–65, 1986.

[MLR+09] Rosa Mora, Alberto López, David Román, Alberto Sendín, and Iñigo Berganza. Communications architecture of smart grids to manage the electrical demand. In *Third Workshop on Power Line Communications*, 2009.

[MLSB10] Rosa Mora, Alberto López, Alberto Sendín, and Iñigo Berganza. Early-stage Smart Grid deployment: leveraging DNO's legacy assets. In *CIRED Workshop, paper*, volume 122, 2010.

[MRN+11] Rosa Mora, R. Ramírez, Isabel NAVALON, CEDETEL-Spain, IBERDROLA-Spain, SIEMENS-Spain, Susana BAÑARES, Pablo MARTIN, Eduardo GARCIA, and REE-Spain. Smart grid communications emulator (100 000 synthetic users). CIRED, 2011.

[NLM+12] Marcel Nassar, Jing Lin, Yousof Mortazavi, Anand Dabak, Il Han Kim, and Brian L Evans. Local utility power line communications in the 3–500 khz band: channel impairments, noise, and standards. *Signal Processing Magazine, IEEE*, 29(5):116–127, 2012.

[ns209] *NS-2*, 2009. `http://www.isi.edu/nsnam/ns/` (Último acceso 13 de julio 2015).

[ns311] *NS-3*, 2011. `https://www.nsnam.org/` (Último acceso 13 de julio 2015).

[omn12] *OMNeT++*, 2012. `https://omnetpp.org/` (Último acceso 13 de julio 2015).

[omn14] *OMNEST*, 2014. `http://omnest.com/` (Último acceso 13 de julio 2015).

[ope11] *Open Meter*, 2011. `http://www.openmeter.com/` (Último acceso 12 de junio 2015).

[opn14] *Riverbed modeler (OPNET)*, 2014. `http://www.riverbed.com/products/performance-management-control/network-performance-management/network-simulation.html` (Último acceso 13 de julio 2015).

[ORD07] *ORDEN ITC/3860/2007, de 28 de diciembre, por la que se revisan las tarifas eléctricas a partir del 1 de enero de 2008.*, 2007. `http://www.boe.es/boe/dias/2007/12/29/pdfs/A53781-53805.pdf` (Último acceso 12 de junio 2015).

[PALB13] Gaetano Patti, Giuliana Alderisi, and Lucia Lo Bello. Performance assessment of the PRIME MAC layer protocol. In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 158–164. IEEE, 2013.

[PHB11] Luis Pedraza, Cesar Hernández, and Dora María Ballesteros. Modelo del desvanecimiento selectivo en frecuencia – model of frequency selective fading. *Revista Facultad de Ingeniería Universidad de Antioquia*, (58):114–122, 2011.

[Phi98] Holger Philipps. Performance measurements of powerline channels at high frequencies. In *Proceedings of the International Symposium on Power Line Communications and its Applications (ISPLC)*, volume 229237, 1998.

[pri06] *PRIME Alliance*, 2006. `http://www.prime-alliance.org/` (Último acceso 20 diciembre 2014).

[PRI08] PRIME Alliance. Prime Specification revision v1.3.6, 2008.

[PRI12] PRIME Alliance. Technology Whitepaper, 2012.

[PVYH03] Niovi Pavlidou, AJ Han Vinck, Javad Yazdani, and Bahram Honary. Power line communications: state of the art and future trends. *IEEE Communications Magazine*, 41(4):34–40, 2003.

[R C13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[RD806] *Real Decreto 809/2006*, 2006. `http://www.boe.es/boe/dias/2006/07/01/pdfs/A24789-24794.pdf` (Último acceso 12 de junio 2015).

[RHL+11] Md Mustafizur Rahman, Choong Seon Hong, Sungwon Lee, Jaejo Lee, Md Abdur Razzaque, and Jin Hyuk Kim. Medium access control for power line communications: An overview of the IEEE 1901 and ITU-T G. hn standards. *Communications Magazine, IEEE*, 49(6):183–191, 2011.

[RKU+11] Kaveh Razazian, Amir Kamalizad, Maher Umari, Qi Qu, Victor Loginov, and Michael Navid. G3-PLC field trials in US distribution grid: Initial results and requirements. In *Power Line Communications and Its Applications (ISPLC), 2011 IEEE International Symposium on*, pages 153–158. IEEE, 2011.

[RP61] WD Ray and AENT Pitman. An exact distribution of the Fisher-Behrens-Welch statistic for testing the difference between the means of two normal populations with unknown variance. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 377–384, 1961.

[SBA+12] Alberto Sendín, Iñigo Berganza, Aitor Arzuaga, Anssi Pulkkinen, and Il Han Kim. Performance results from 100,000+ PRIME smart meters deployment in Spain. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 145–150. IEEE, 2012.

[SBA+13] Alberto Sendín, Iñigo Berganza, Aitor Arzuaga, Xabier Osorio, Iker Urrutia, and Pablo Angueira. Enhanced operation of electricity distribution grids through smart metering PLC network monitoring, analysis and grid conditioning. *Energies*, 6(1):539–556, 2013.

[SC89] George W Snedecor and Witiiam G Cochran. Statistical methods 8. ed. *Iowa State Univ*, 1989.

[Sch09] Mischa Schwartz. Carrier-wave telephony over power lines: Early history [history of communications]. *Communications Magazine, IEEE*, 47(1):14–18, 2009.

[SGS10] B. Sivaneasan, E. Gunawan, and P.L. So. Modeling and performance analysis of automatic meter-reading systems using PLC under impulsive noise interference. *Power Delivery, IEEE Transactions on*, 25(3):1465–1475, 2010.

[SLAB11a] Alberto Sendín, Asier Llano, Aitor Arzuaga, and Iñigo Berganza. Strategies for PLC signal injection in electricity distribution grid transformers. In *Power Line Communications and Its Applications (ISPLC), 2011 IEEE International Symposium on*, pages 346–351. IEEE, 2011.

[SLAB11b] Alberto Sendín, Asier Llano, Aitor Arzuaga, and Iñigo Berganza. Field techniques to overcome aggressive noise situations in PLC networks. In *Power Line Communications and Its Applications (ISPLC), 2011 IEEE International Symposium on*, pages 113–117. IEEE, 2011.

[SPMG12] A. Sanz, P.J. Pinero, D. Montoro, and J.I. García. High-accuracy distributed simulation environment for PRIME networks analysis and improvement. In *Power Line Communications and Its Ap-*

plications (ISPLC), 2012 16th IEEE International Symposium on, pages 108–113. IEEE, 2012.

[SPMG13] A. Sanz, P.J. Pinero, S. Miguel, and J.I. García. Real problems solving in PRIME networks by means of simulation. In *Power Line Communications and Its Applications (ISPLC), 2013 17th IEEE International Symposium on*, pages 285–290. IEEE, 2013.

[SPMGN12] A. Sanz, P.J. Pinero, S. Miguel, and J.I. García-Nicolás. Distributed event-driven simulation environment for smart metering protocols evaluation. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 151–156. IEEE, 2012.

[Tel01] Telecommunications Industry Association et al. Tia/eia-568-b commercial building telecommunications cabling standard, 2001.

[TV10] Andrea M Tonello and Fabio Versolatto. Bottom-up statistical PLC channel modeling—part II: inferring the statistics. *Power Delivery, IEEE Transactions on*, 25(4):2356–2363, 2010.

[TXB+07] Zheng Tao, Yang Xiaoxian, Zhang Baohui, Nucleus H Xu, Feng Xiaoqun, and Li Changxin. Statistical analysis and modeling of noise on 10-kV medium-voltage power lines. *Power Delivery, IEEE Transactions on*, 22(3):1433–1439, 2007.

[UPC15] Centro de Innovación y Tecnología UPC. *Smart Grids*, 2015. `http://cit.upc.edu/es/destacados/smart_grid` (Último acceso 18 octubre 2015).

[VH08] András Varga and Rudolf Hornig. An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[ZD00] M. Zimmerman and K. Dostert. The low voltage power distribution network as last mile access network–signal propagation and noise scenario in the HF-Rang. *International Journal of Electronics and Communications (AEU)*, 54(1):13–22, 2000.

[ZD02a] Manfred Zimmermann and Klaus Dostert. Analysis and modeling of impulsive noise in broad-band powerline communications. *Electromagnetic Compatibility, IEEE Transactions on*, 44(1):249–258, 2002.

[ZD02b] Manfred Zimmermann and Klaus Dostert. A multipath model for the powerline channel. *Communications, IEEE Transactions on*, 50(4):553–559, 2002.

Esta tesis fue terminada de escribir en Bilbao el 14 de diciembre de 2015