Universidad Rey Juan Carlos

Máster en Redes y Servicios de Comunicación Móviles

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE MÁSTER

DISEÑO DE UN PROTOCOLO JERÁRQUICO DE COMUNICACIONES PARA ENCAMINAMIENTO DE PAQUETES EN REDES DE SENSORES INALÁMBRICOS

Autor: Francisco Javier Atero Gómez

Tutor: Juan José Vinagre Díaz

Curso académico 2007/2008

Universidad Rey Juan Carlos

ACTA DE EVALUACIÓN

Alumno:

Titulación:

Título del proyecto:

¿Es el proyecto resultado de Prácticas en empresas? SÍ / NO

Tutor:

Co-Tutor (en caso de que exista):

TRIBUNAL

Presidente:

Vocal:

Secretario:

CALIFICACIÓN DETALLADA DEL PROYECTO

	Presidente	Vocal	Secretario
Presentación escrita (MB-B-R-M-MM) **Presentación oral** (MB-B-R-M-MM)			
Complejidad técnica (MB-B-R-M-MM)			
Metodología empleada (MB-B-R-M-MM)			
Resultados obtenidos (MB-B-R-M-MM)			
Esfuerzo realizado (MB-B-R-M-MM)			

CALIFICACIÓN FINAL DEL PROYECTO:

	SB/ NOT/ AP/ SS/ NP
(nota numérica)	Enmarcar la calificación alcanzada

PROYECTO PROPUESTO PARA MATRÍCULA DE HONOR: SÍ/NO
(sólo si la nota numérica final es igual a 9,5)

Fuenlabrada, de de 20__

El Presidente **El Vocal** **El Secretario**

Universidad Rey Juan Carlos

Agradecimientos

No hubiera sido posible la realización de este Proyecto Fin de Máster sin la generosa colaboración de multitud de personas a las que quería expresar mi agradecimiento por su ayuda no sólo en el desarrollo y en la confección de este trabajo.

Sirvan por tanto estas líneas como testimonio de gratitud y reconocimiento a la labor realizada por mi amigo y tutor, Juan José Vinagre, cuyo apoyo y atención ha sido una fundamental para mí a lo largo de estos últimos meses; a Mark Wilby, por servir de guía y por su visión, disposición y ayuda en todo lo relacionado con el diseño teórico; a Antonio Caamaño, por su colaboración presente y futura en este proyecto; a Javier Ramos la confianza depositada y en especial a todos los integrantes del Departamento de Teoría de la Señal y Comunicaciones de la Universidad Rey Juan Carlos y del CECI del Departamento por su apoyo y acogida en este último año.

Universidad Rey Juan Carlos

Resumen

A lo largo de los últimos años las Redes de Sensores Inalámbricos (RSI) se han convertido en foco de atención de la comunidad científica por su enorme potencial en el desarrollo de servicios basados en la monitorización de variables del entorno físico. Estas redes, que son un caso particular de las Redes Ad Hoc Inalámbricas, son capaces de proporcionar soluciones en cuanto al establecimiento de redes autónomas que no tengan necesidad de una infraestructura externa para su operación.

Para que estas redes puedan cumplir con los objetivos propuestos de recogida y transmisión de la información de su entorno, se han desarrollado en los últimos años una gran cantidad de protocolos de encaminamiento de muy diversa índole, en función del tipo de RSI concreta en el que se quiera aplicar.

Una característica especialmente importante de los protocolos de encaminamiento en este tipo de redes es que deben poder adaptarse rápidamente a los cambios continuos de la red con el fin de mantener un conocimiento actualizado de las rutas disponibles entre cualquier pareja de nodos de la misma.

El presente Proyecto Fin de Máster aborda precisamente el estudio del encaminamiento en RSI, fijando como objetivo la propuesta de un algoritmo de encaminamiento jerárquico que tenga en consideración los mensajes que transitan por la red desde su pasarela central hacia los nodos sensores, de forma que se optimicen las prestaciones de la RSI debidas a este aspecto. Para ello se analizan diferentes escenarios y se seleccionan las alternativas más adecuadas en cada caso para tratar de optimizar la eficiencia del mismo incluso en las condiciones más desfavorables.

Índice

1.- Introducción y antecedentes

1.1.- Descripción de la necesidad

Las Redes de Sensores Inalámbricos (RSI) se han convertido en foco de atención de la comunidad científica en los últimos años. Este desarrollo en el plano de la investigación ha venido acompañado de unos primeros pero firmes pasos en el ámbito comercial que han reforzado la idea de la necesidad de este tipo de redes para innumerables aplicaciones que necesitan de una monitorización del entorno para desarrollar servicios en multitud de ámbitos dispares como la seguridad física, la sanidad o la agricultura. Este gran abanico de soluciones se encuadra dentro de la llamada *inteligencia ambiental* (AmI, *Ambient Intelligence*) [1], que promueve la creación de una interacción constante, directa y transparente del usuario con su medio circundante de forma que sea posible la oferta de servicios personalizados a las características particulares del mismo: hábitos, localización, preferencias, etc.

Desde el punto de vista tecnológico, estas RSI son, en definitiva, un subconjunto de la categoría superior formada por las Redes Ad Hoc Inalámbricas (RAHI), capaces de proporcionar soluciones a las exigencias actuales relativas al establecimiento de redes autónomas que no presentan restricciones en cuanto a la necesidad de una infraestructura externa para su operación. Este grado de autonomía es conseguido mediante la capacidad de las RAHI de autoconfigurarse y admitir actualizaciones dinámicas en tiempo y espacio.

Evidentemente, estas fuertes restricciones suponen un gran reto tecnológico que deberá superar los problemas subyacentes a las principales características intrínsecas de las RAHI: variabilidad en la topología y el canal y ausencia de infraestructura. Estas limitaciones inciden en la mayor parte de los aspectos involucrados en la operación de una red de comunicaciones. Entre ellos, uno de los más afectados es el encaminamiento, al mismo tiempo, crucial para el funcionamiento de una RAHI y las prestaciones ofrecidas por ella. En particular, en las RSI el encaminamiento es responsable de variables

fundamentales para la misma como el consumo energético, el retardo o la tasa de error de bit. Estas variables inciden directamente en la calidad de servicio (QoS, *Quality of Service*) ofrecida e incluso en la vida útil de la red por lo que deben ser tomadas en profunda consideración durante la investigación y desarrollo de esta nueva tecnología.

Una RSI se estructura habitualmente en forma de estrella en la que el nodo que actúa de pasarela (*gateway*) entre la propia red y el servidor central, se configura como el centro de la misma, y los nodos sensores ocupan posiciones en sus brazos siguiendo una organización jerárquica. Así, las comunicaciones en una RSI se establecen usualmente desde estos nodos sensores hacia la pasarela central, mediante un esquema multisalto en el que los dispositivos intermedios de un camino hacen las veces de repetidores o reencaminadores hasta que el mensaje alcanza el destino final. Teniendo en cuenta este esquema de transporte de información, el encaminamiento diseñado para una RSI se centra precisamente en el sentido nodo sensor – pasarela central, construyendo algoritmos jerárquicos que optimizan por una parte la distancia de la ruta resultante y, por otra, su fiabilidad. Sin embargo, los paquetes enviados en el sentido opuesto, pasarela central – nodo sensor, son encaminados simplemente mediante inundación. Este modo de operación podría ser admisible en escenarios en los que se obvie el control y la gestión de la RSI. Sin embargo, éste es un requerimiento cada vez más acuciante debido al tamaño y funcionalidades esperados para las mismas. Bajo estas premisas, el encaminamiento por inundación de los mensajes de control procedentes de la pasarela central generaría un volumen de datos que reduciría ostensiblemente las prestaciones de la RSI, pudiendo llegar incluso a colapsarla.

El presente Proyecto Fin de Máster (PFM) aborda precisamente el estudio del encaminamiento en RSI, fijando como objetivo la propuesta de un algoritmo de encaminamiento jerárquico que tenga en consideración los mensajes que transitan por la red desde su pasarela central hacia los nodos sensores, de forma que se optimicen las prestaciones de la RSI debidas a este aspecto.

1.2.- Redes Ad Hoc inalámbricas y Redes de Sensores Inalámbricos

El origen de las RAHI puede fijarse en los años 70, cuando la *Defense Advanced Research Projects Agency* (DARPA) desarrolló la llamada Red Radio de Paquetes (*Packet Radio Network*) para comunicaciones entre vehículos en movimiento. Ésta fue la primera experiencia de redes carentes de infraestructura cableada, semilla de las actuales RAHI.

Los avances ulteriores de la tecnología electrónica han hecho posible la integración de terminales y dispositivos de red en un único nodo ad hoc, y la interconexión inalámbrica entre los mismos. El primer registro de la investigación generada en este ámbito data de 1995 y pertenece a la IETF (*Internet Engineering Task Force*) [2], que crea oficialmente el Grupo de Trabajo en RAHI móviles (MANET-WG, *Mobile Ad hoc NETworks – Working Group*). Esta misma organización, en su recomendación RFC2501, adopta la ya establecida en el estándar 802.11 del IEEE (*Institute of Electrical and Electronics Engineers*) [3] donde se contempla la configuración de operación independiente de las estaciones, ad hoc, en la capa de Control de Acceso al Medio (MAC), de tal forma que sea posible la comunicación directa entre ellas, añadiendo la capacidad de movilidad de los nodos. Finalmente, en diciembre de 2003, se creó en Japón el Consorcio de Redes Ad Hoc (*Ad Hoc Network Consortium*) con el objetivo de unir los intereses y esfuerzos de la industria, la comunidad científica y las autoridades públicas para el fomento y desarrollo de las RAHI en los distintos ámbitos de aplicación a los que puede dar respuesta.

En su vertiente tecnológica, este desarrollo de las RAHI se concreta en acciones de estandarización de distintas implementaciones de las mismas. Así aparece el IEEE 802.11s, que incluye la definición de las llamadas *redes malladas* (*mesh networks*), que presentan la particularidad de habilitar el establecimiento de comunicaciones a través de mecanismos multisalto sobre redes autoconfigurables. La versión D0.01 de dicho estándar fue admitida como tal en marzo de 2006 y en la actualidad se encuentra en su versión D1.00.

Por su parte, el subconjunto de RAHI dedicado a las RSI también dispone de un estándar generado por el grupo de trabajo del IEEE dedicado a las Redes de Área Personal

(PAN, *Personal Area Networks*), el que además dedica esfuerzos orientados en el sentido de las redes malladas (802.15.5). Las RSI en particular quedan recogidas en la especificación IEEE 802.15.4, en las llamadas PAN de baja tasa. La última versión del estándar, 802.15.4-2006, apareció en septiembre de 2006. El estándar 802.15.4 recoge las capas física (PHY) y MAC y está respaldado por la Alianza ZigBee, que cuenta entre sus promotores con empresas como Philips, Honeywell, Siemens, Texas Instruments, Schneider Electric, Samsung, Motorola, Mitsubishi o Freescale. La certificación de productos ZigBee además incluye aspectos relacionados con capas superiores no incluidas en el 802.15.4 considerando como requerimientos esenciales la minimización de la complejidad, el coste y el consumo energético.

A continuación se especifican algunas de las características de operación de las RAHI:

Capa física

Los nodos pertenecientes a una red inalámbrica se comunican entre sí utilizando como medio de transmisión el espacio libre, haciendo uso de canales de radiofrecuencia (RF). Los canales de RF presentan una serie de problemas como la atenuación de la señal o el desvanecimiento debido al multitrayecto, que limitan las prestaciones en cuanto a QoS de las RAHI.

Para la implementación de estas comunicaciones inalámbricas, los nodos utilizan frecuentemente antenas omnidireccionales que permiten su comunicación con cualquier otro dispositivo dentro de su rango de cobertura. El uso de este tipo de antenas influye en la velocidad de transmisión de las RAHI ya que, durante las fases de transmisión o recepción de un nodo en particular, el resto de los nodos pertenecientes a su vecindario deben permanecer inactivos, lo que imposibilita la ocupación de la totalidad de la capacidad de la red.

Los estándares IEEE 802.11x definen interfaces para canales de RF en las bandas de 2.4 y 5 GHz, siendo la primera la más extendida. Aun gozando de todas las ventajas provenientes de su carácter no licenciado, esta banda presenta cierta saturación al ser muchas las tecnologías que optan por utilizarla para aplicaciones totalmente dispares, lo que conlleva una baja fiabilidad en este sentido

Capa MAC

Existen dos categorías principales de protocolos MAC: los protocolos de acceso aleatorio, en los cuales los nodos compiten entre sí para ganar el acceso al medio de transmisión compartido, y los protocolos de acceso controlado, en los que un nodo maestro o de infraestructura decide cuál de los nodos puede acceder al medio de transmisión en cada momento. La falta de una infraestructura y la naturaleza mallada de las RAHI hacen que los protocolos de acceso aleatorio sean la opción natural para el control del acceso al medio en este tipo de redes.

Algunos ejemplos de los protocolos MAC utilizados en las RAHI son los siguientes: MACA (*Multiple Access with Collision Avoidance*) [4], MACA-BI (MACA *by Invitation*) [5] y FAMA (*Floor Acquisition Multiple Access*) [6]. De entre las diversas propuestas, el Comité IEEE 802.11 eligió a CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), una variante de MACA, como la base para sus estándares, debido a su inherente flexibilidad.

Entre los protocolos MAC de acceso controlado se encuentran TDMA (*Time Division Multiple Access*), FDMA (*Frequency Division Multiple Access*), CDMA (*Code Division Multiple Access*) y TSMA (*Time Spread Multiple Access*). Aunque estos protocolos raramente se utilizan en RAHI, son seleccionados para escenarios en los que la garantía de QoS es una necesidad ya que sus transmisiones están libres de colisiones. Su aplicación está principalmente adaptada a redes como Bluetooth y a otras RAHI basadas en grupos de nodos (*clusters*), en las cuales el acceso al medio

está controlado por nodos maestros que deciden qué nodo del grupo es el que tiene acceso al canal, posibilitando así transmisiones libres de contienda y colisión.

Capa de Red

Una característica especialmente importante de los protocolos de encaminamiento para RAHI es que deben poder adaptarse rápidamente a los cambios continuos de la red con el fin de mantener un conocimiento actualizado de las rutas disponibles entre cualquier pareja de nodos de la misma.

Estos protocolos serán estudiados en detalle en el *Apartado 1.3.1 Encaminamiento en RAHI y RSI.*

Capa de transporte

Las RAHI tienen como objetivo final la implantación de la llamada Internet inalámbrica omnipresente y ubicua, por lo que está basada en la pila de protocolos TCP/IP (*Transmisión Control Protocol / Internet Protocol*). Esto ha significado que prácticamente todos los esfuerzos de investigación y desarrollo del MANET-WG estén principalmente dirigidos al desarrollo de algoritmos de encaminamiento especialmente adecuados para este tipo de redes, sin que los demás protocolos de la pila se vean afectados. Aun así, en los últimos años se han presentado varias propuestas encaminadas a mejorar las prestaciones de los principales protocolos de transporte de la Internet, TCP y UDP (*User Datagram Protocol*), en entornos ad hoc.

Entre estas mejoras está el empleo de mecanismos de señalización explícita que permitan tener un TCP "amigable" con las RAHI, para evitar la activación errónea del mecanismo de control de congestión ante la pérdida de la ruta entre los nodos fuente y destino debida a un fallo en el enlace [7]. Otros autores han propuesto la creación de TPA (*Transport Protocol for Ad hoc Networks*) [8], un nuevo protocolo de transporte especialmente diseñado para entornos ad hoc, que incluye

mecanismos para detección de pérdida del enlace y recuperación de ruta, además de establecer un mecanismo de control de congestión diferente al de TCP.

Los protocolos de transporte SCTP (*Stream Control Transfer Protocol*) [9] y el MRTP (*Multiflow Realtime Transport Protocol*) [10] han sido especialmente diseñados para ofrecer un mejor transporte de datos a las aplicaciones de difusión de audio y vídeo y de tiempo real.

Capa de aplicación

Uno de los objetivos del MANET-WG es que las mismas aplicaciones diseñadas para Internet puedan funcionar en las RAHI. Esto es válido actualmente para las aplicaciones de transmisión de datos, que pueden aceptar una entrega de información bajo la filosofía denominada *best effort*, sin grandes requerimientos de ancho de banda ni restricciones en cuanto al tiempo de entrega de los paquetes. Sin embargo, Internet también es utilizada para otros tipos de aplicaciones, como aquéllas que permiten el acceso en tiempo real a contenidos de audio y vídeo en repositorios remotos que producen flujos continuos de datos (*streams*), así como aplicaciones multimedia interactivas de tiempo real, como la voz sobre IP, el vídeo sobre IP, la videoconferencia y la videotelefonía, que demandan a la red de transporte ciertas garantías mínimas de retardo en la entrega de paquetes, de variación de este retardo y de ancho de banda, pero que, por otra parte, pueden soportar cierto nivel de pérdida de paquetes, sin incurrir en una ausencia de utilidad de la información recibida.

Debido a que el ancho de banda y la fiabilidad de los enlaces de las RAHI son mucho menores que en las redes cableadas, hay un gran número de retos por resolver para que este tipo de aplicaciones pueda ser soportado de manera adecuada en estos escenarios. Algunas propuestas se centran en disminuir al mínimo el ancho de banda requerido por las aplicaciones, basándose en nuevas técnicas de codificación y compresión de la información, que sacrifican la calidad de

la información para todos los nodos y condiciones de la red por igual. Una alternativa que goza de una creciente aceptación es la que consiste en capacitar a las aplicaciones para su adaptación a las condiciones dinámicas de las RAHI y a las prestaciones particulares de los nodos en ambos extremos.

1.3.- Encaminamiento en RAHI y RSI

1.3.1.- Encaminamiento en RAHI

Sobre las bases expuestas anteriormente, el encaminamiento en RAHI debe satisfacer ciertos objetivos [11]:

- *Minimización de costes asociados*: reducción del número de mensajes de control y la carga computacional en los nodos para tareas de encaminamiento, de forma que se optimice el uso de recursos escasos como energía y ancho de banda.
- *Capacidad multisalto*: reencaminamiento de información a través de nodos intermedios de una ruta construida entre fuente y destino.
- *Mantenimiento dinámico de topología*: actualización de frecuencia suficiente de las rutas a seguir entre cualquier par de nodos de la red, adaptándose a los cambios topológicos inherentes a este tipo de redes.
- *Eliminación de bucles*: anulación de la posibilidad de segundo paso por un mismo nodo.

Además, según la recomendación RFC2501 de la IETF [2], el encaminamiento en RAHI debe admitir diversos modos de operación:

- *Distribuido*: cada nodo se ocupa de las tareas necesarias de encaminamiento de forma local e independiente.

- *Bajo demanda*: la red adapta el encaminamiento utilizado a la situación específica actual representada por su patrón de tráfico; este modo de operación optimiza el consumo de energía y ancho de banda, pero introduce cierto retardo debido a la necesidad de cálculo de ruta actualizado.

- *Activo*: los nodos de la red implementan un envío sistemático de información a través de rutas previamente trazadas.

- *De letargo*: ante una ausencia de actividad, los nodos conmutan a un estado de baja energía y funcionalidades restringidas.

Siguiendo los requisitos expuestos anteriormente, la comunidad científica ha propuesto en los últimos años una gran cantidad de protocolos de encaminamiento para las RAHI. Dada la extensión de este conjunto de protocolos desarrollados, pueden encontrarse a su vez distintas categorizaciones como las presentes en [12], [13], [14], [15] y [16], que los agrupan según ciertos criterios sobre modo de operación o funcionalidad. En el presente PFM, se incluye la taxonomía propuesta en la tesis doctoral en [17], que queda resumida en la siguiente figura:

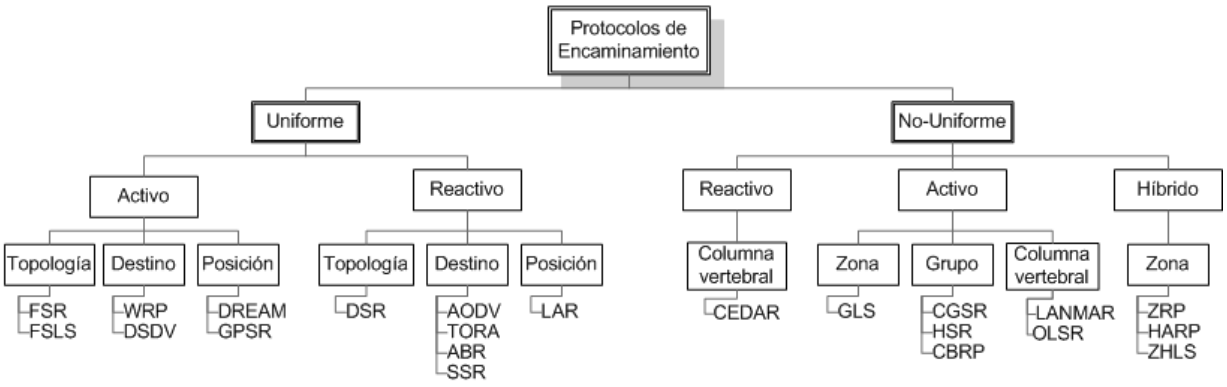


Figura 1. Taxonomía de protocolos de encaminamiento en RAHI

La **clasificación principal** se refiere a la homogeneidad o heterogeneidad de los nodos en cuanto a su funcionalidad en el ámbito del encaminamiento, distinguiéndose entre:

- *Uniforme*: o de estructura plana. Las funcionalidades de todos los nodos relativas al encaminamiento son iguales. La red se estructura de forma independiente de sus nodos al poseer todos ellos las mismas capacidades. Este planteamiento conlleva una minimización de los costes energético y de ancho de banda requeridos para llevar a cabo el mantenimiento dinámico de la topología de la red. Por el contrario, no es capaz de ofrecer un alto nivel de escalabilidad, degradando sus prestaciones a medida que crece el tamaño de la red.

- *No uniforme*: la red presenta una estructura jerárquica, compuesta por distintos niveles de funcionalidad que pueden conllevar diferentes capacidades de cómputo, energía o almacenamiento en un subconjunto de nodos. Bajo esta premisa, se hace más complejo el mantenimiento dinámico de la estructura de la red mientras que se consigue una mayor escalabilidad y un mejor aprovechamiento de los recursos escasos como energía y ancho de banda.

El **segundo nivel de clasificación** se refiere al modo de operación seguido en la construcción de las rutas que serán utilizadas para la comunicación entre distintos pares de nodos fuente-destino. Así, se dividen en:

- *Activo*: para implementar las transmisiones, cada nodo utiliza una tabla previamente construida en la fase llamada *de descubrimiento de ruta*, donde se indica el camino que debe seguir, bien de forma local o bien de forma completa. De esta forma, se consigue minimizar el retardo en la transmisión al obviar el cálculo de la ruta para la misma. Sin embargo, este procedimiento necesita la actualización periódica de las tablas, lo que puede suponer una gran cantidad de mensajes de control en escenarios de alta variabilidad, que generen un consumo de recursos de red relativos a energía, cómputo o almacenamiento inaceptable en

algunas aplicaciones y, además, independientemente del grado de utilización de cada ruta.

- *Reactivo*: también denominado *bajo demanda*'. Antes de efectuar cada transmisión, los nodos desencadenan un proceso de búsqueda de ruta hasta el destino requerido. Al contrario que en el modo de operación anterior, este procedimiento resulta en una disminución significativa del coste del mantenimiento de caminos, introduciendo no obstante un retardo en la transmisión así como un uso ineficiente de la capacidad de la red debido a la inundación de la misma con los mensajes de control necesarios para los frecuentes procesos de descubrimiento de ruta.

- *Híbrido*: utilizado específicamente por protocolos no uniformes. Este modo de operación utiliza uno de los dos anteriores en los distintos niveles jerárquicos definidos. En consecuencia, se llega a un compromiso entre la latencia incluida por los procesos de descubrimiento de rutas y la generación de mensajes de control para el mantenimiento de los caminos construidos.

El **último nivel de clasificación** es específico de cada grupo de protocolos definido por los dos niveles anteriores. Así, los protocolos uniformes se subdividen dependiendo del tipo de información sobre el estado de la red, que manejan los nodos para efectuar las labores de mantenimiento. Bajo este criterio, los protocolos uniformes, tanto activos como reactivos, se clasifican como:

- *Topología*: la información manejada por los nodos es de ámbito global a la red. Un conjunto extenso de estos protocolos se basan en la diseminación a toda la red de la información sobre las conexiones disponibles entre cada nodo y su vecindario. De esta forma, todos los nodos tienen un conocimiento preciso del grafo de la red y, por tanto, son capaces de encaminar sus mensajes atendiendo a las rutas de menor longitud. Sin

embargo, esta aproximación no arroja grandes resultados a medida que se incrementa la variabilidad de la red y el consiguiente aumento drástico de los mensajes de control necesarios para mantener la información topológica de la misma actualizada.

- *Destino*: el ámbito de la información que poseen los nodos es local. La mayor parte de los protocolos pertenecientes a este grupo son los llamados *distancia-vector*, que implementan las labores de encaminamiento sobre el conocimiento de cierta medida de distancia y el vector de dirección hasta el destino final, usualmente expresados como el número de saltos mínimo hasta el mismo y el siguiente salto a realizar, respectivamente.

- *Posición*: una vez más, el ámbito de la información presente en los nodos es global, en este caso con la peculiaridad de fundamentarse en las coordenadas geográficas de los nodos de la red. Sobre la base de este conocimiento, el encaminamiento se implementa en forma de aproximaciones secuenciales hacia el destino según un criterio de maximización de la componente de distancia de cada salto en la dirección de la línea recta que une fuente y destino. Esta técnica produce una gran eficiencia en escenarios cercanos al estático, mientras que en situaciones de alta variabilidad, discontinuidades u obstáculos, necesita del uso de algoritmos específicos para mejorar su rendimiento. Por supuesto, la mayor restricción de este grupo de protocolos es el requerimiento de un sistema de posicionamiento, absoluto o relativo, para su operación.

Por su parte, los protocolos no uniformes pueden ser catalogados según el tipo de organización que se sigue para la estructuración de la red en distintos niveles jerárquicos. Así, pueden diferenciarse en:

- *Zona*: la agrupación de los nodos atiende a criterios geográficos, reduciendo de esta forma los mensajes de control para el mantenimiento de dicha estructura al ámbito local de la misma. Para hacer efectivo este encaminamiento, es una vez más necesario un sistema de posicionamiento que permita el conocimiento de las ubicaciones de los nodos para su agrupación.

- *Grupo*: los nodos se reúnen en torno a uno en particular, *clusterhead*, que desempeña labores de coordinación en distintos aspectos entre los que se encuentra el encaminamiento. Esta agrupación es en principio lógica y no tiene por qué atender a criterios geográficos como en la categoría anterior. La principal ventaja de este esquema es la reducción que implica en cuanto a los mensajes de control, que se restringen al ámbito local de cada grupo. Por el contrario, los nodos que actúan como coordinadores necesitan frecuentemente una serie de capacidades superiores a las del resto.

- *Columna vertebral*: en este caso, no sólo un nodo sino un conjunto de ellos son seleccionados dinámicamente para desempeñar funciones especiales en la construcción de caminos y la generación y reenvío de mensajes de control y datos. Este grupo de nodos es denominado *columna vertebral* (*backbone*) de la red. La red en hace uso de esta columna vertebral para implementar el encaminamiento requerido en cada caso. De esta forma, se consigue la escalabilidad de la red y una reducción del coste referido al control del encaminamiento, incurriendo, sin embargo, en gastos relacionados al mantenimiento de la estructura jerárquica.

1.3.2.- Encaminamiento en RSI

Como subconjunto de las RAHI, las RSI adoptan las generalidades en cuanto a requisitos de aplicación para el encaminamiento descritas en el apartado anterior.

La primera peculiaridad de las RSI es el propio esquema de comunicaciones en una estructura de estrella, basado en transmisiones entre los nodos sensores y la pasarela o estación base. Sobre este grupo de necesidades, las RSI añaden una nueva serie de restricciones particulares procedentes tanto de su propia especificación recogida en el estándar 802.15.4 –bajo consumo de energía, baja tasa binaria, bajo coste y baja complejidad–, como de las aplicaciones o formas de funcionamiento de estas redes. Entre ellas, cabe destacar las siguientes:

- *Despliegue* de los nodos: puede ser tanto arbitrario como prediseñado. El nivel de arbitrariedad del despliegue redunda directamente en la complejidad de los procedimientos necesarios para maximizar la eficiencia del encaminamiento.

- *Consumo energético*: esta limitación incide sobre la carga computacional que se permite para el encaminamiento así como el número de retransmisiones aceptables según la precisión requerida.

- *Modelo de envío de datos*: puede ser continuo, generado por un suceso, generado por una petición o híbrido.

- *Tolerancia a fallos*: la red debe adaptarse a los fallos que pudieran surgir en algunos de sus nodos, reconfigurando su encaminamiento para continuar las comunicaciones dentro de los límites establecidos.

- *Escalabilidad*: el encaminamiento debe estar preparado para soportar una gran cantidad de nodos así como aumentos puntuales en su tamaño, sin incurrir por ello en reducciones de las prestaciones conseguidas.

- *Movilidad*: aunque en una gran cantidad de aplicaciones los nodos sensores tendrán un carácter estático, se debe contemplar también la posibilidad de movilidad de la totalidad o un subconjunto de los nodos de la red. Estos cambios dinámicos de la topología de la red harán que el encaminamiento deba considerar el grado de movilidad de la misma de

forma que se seleccionen los protocolos que mejor se adapten al mismo en función de los requisitos de calidad especificados.

- *Conectividad*: en principio, una RSI se concibe como una red densa y por tanto, cada nodo tendrá una alta probabilidad de encontrar al menos un nodo vecino dentro de su radio de acción. Sin embargo, es necesario tener en consideración la posible aparición de lagunas de conectividad y enlaces críticos que unan secciones de red diferenciadas y que, por tanto, deban soportar una mayor tasa de datos a su través.

- *Agregación de datos*: dada la densidad supuesta para este tipo de redes, en la mayor parte de los casos los datos generados presentarán una alta redundancia. Este hecho deberá ser tenido en cuenta dado que podría permitir agregación de datos, evitando transmisiones innecesarias y, por tanto, un ahorro significativo en el consumo de energía.

- *Calidad de servicio*: el encaminamiento deberá tener en cuenta el grado de QoS según las variables definidas para la misma para seleccionar la técnica que mejor se adapte a la consecución de los objetivos marcados.

A continuación, se describen algunos de los protocolos de encaminamiento utilizados en RSI según distintas clasificaciones en función de su homogeneidad (uniforme, no uniforme), el conocimiento sobre la posición de los nodos y su forma de operación (multicamino, por petición, por negociación, por calidad de servicio y coherente/no coherente).

Por estructura

- *Uniforme*

  La gran cantidad de sensores presente en estas redes hace que la asignación de un identificador global a cada nodos sea cuando menos no recomendable. Esta consideración conduce a que la recogida de datos se lleve a cabo mediante peticiones realizadas desde una

estación base a un área determinada. Los protocolos de encaminamiento se centran por tanto en negociación y eliminación de datos redundantes.

- *Protocolos basados en negociación* [18]. También llamados SPIN (*Sensor Protocols for Information via Negotiation*). Diseminan la información de cada nodo al resto de la red, asumiendo que todos son estaciones base potenciales. Antes del envío de datos, se realiza una caracterización de los mismos en meta-datos, que son negociados con los nodos vecinos para asegurar la eliminación de redundancia. Como desventaja se puede destacar que el protocolo no garantiza la entrega de los datos.

- *Difusión directa* [19]. Propone un paradigma de agregación de datos en el los mismos son clasificados bajo diferentes valores en cada nodo y sus vecinos. El envío de la información hacia la estación base se produce cuando ésta transmite la petición de datos de una característica determinada a los nodos vecinos y éstos a su vez a los suyos, que responden con los gradientes solicitados. De esta forma, se establecen rutas específicas para cada tipo de datos. Esta técnica no es apropiada para los casos en los que el envío de datos es continuo ya que el modelo de petición de datos no resulta conveniente. Por otra parte, el hecho de que los sensores elijan los datos según la petición requiere mayor capacidad de computación en los nodos.

- *Encaminamiento por rumor* [20]. Variante del anterior en la que se encaminan peticiones sólo a los nodos que han registrado un suceso en particular en lugar de a toda la red. Se emplean paquetes llamados *agentes*, que son

creados cuando un nodo registra un suceso. Ante una petición, sólo responden los nodos que tienen el agente en su tabla de sucesos. Sólo se mantiene un camino entre la fuente y el destino. Esta técnica obtiene buenos resultados en situaciones en las que el número de sucesos es pequeño ya que en el resto de los casos, sería muy costoso el mantenimiento de las tablas de sucesos y los agentes.

- *Minimización del coste de reenvío* [21]. Se define el algoritmo MCFA (*Minimum Cost Forwarding Algorithm*), que se basa en el conocimiento por parte de cada nodo de la estimación de menor coste desde sí mismo hasta la estación base. De esta forma, cuando le llega un paquete, observa si está en el camino de menor coste y, si es así, lo propaga a sus vecinos.

- *Encaminamiento basado en gradiente* [22]. Propone una variante de la Difusión Directa. Se basa en memorizar el número de saltos cuando se difunde el *interés* por la red en Difusión Directa, calculando la llamada *altura* del nodo (mínimo número de saltos para alcanzar la estación base). Se entiende como *gradiente* la diferencia de alturas entre un nodo y su vecino.

- *No uniforme*

Los protocolos jerárquicos o basados en *clusters* ofrecen claras ventajas relacionadas con la escalabilidad y la eficiencia en la comunicación.

- *Jerarquía de baja energía adaptativa* [23]. Conocidos como LEACH (*Low Energy Adaptive Clustering Hierarchy*).

Selecciona unos *clusterheads* de forma arbitraria que recogen la información generada por los nodos de su grupo y la envían agregada a la estación base. Resulta muy apropiado para las situaciones en las que se necesite un envío continuo de datos. Como contrapartida, LEACH asume que todos los nodos pueden alcanzar la estación base, premisa que no es admisible en redes de gran tamaño.

- *PEGASIS* [24] (*Power-Efficient Gathering in Sensor Information Systems*). Mejora las prestaciones de LEACH creando turnos de comunicación de cada nodo en un *cluster* a la estación base y permitiendo tan sólo comunicaciones entre un nodo y su vecino y la estación base. Consigue aumentar la vida útil de la red. Sin embargo, asume que todos los nodos mantienen una base de datos de localización del resto sin especificar el método para lograrlo. A su vez, como LEACH, da por sentado que todos los nodos pueden contactar con la estación base, lo que no siempre es viable.

- *Encaminamiento basado en umbral* (TEEN [25] y APTEEN [26]). Se proponen para aplicaciones en las que el tiempo sea un parámetro crítico. En TEEN los sensores miden el medio constantemente, pero sólo transmiten cuando observan una magnitud por encima de un determinado límite o cuya variación en relación con la medida actual está por encima de un segundo límite. Estos límites les han sido enviados por el *clusterhead*. APTEEN por su parte cambia la periodicidad o los valores límite de acuerdo con las necesidades del usuario o la aplicación. El *clusterhead* envía los *atributos* (magnitudes de interés),

límites (los mismos que en TEEN), *horario* (que asigna una ranura de tiempo a cada nodo) y *contador de tiempo* (máximo período entre dos envíos sucesivos). Como desventajas, mencionar el aumento de información en la red y la complejidad necesaria para gestionar los límites.

- *Regiones de minima energía* [27]. En MECN (*Minimum Energy Communication Network*) se identifica una región para cada nodo en la que la transmisión multicamino es más eficiente que la transmisión directa en términos de consumo energético.

- *Estaciones base móviles* [28]. En este caso, TTDD (*Two-Tier Data Dissemination*) se basa en nodos agrupados en *clusters* que envían los datos recogidos después de un suceso a unas estaciones base móviles. Los nodos están emplazados en una celda que se construye a priori desde la estación base. El problema reside en el método para la localización de los nodos necesario para la construcción de la cuadrícula de celdas.

Por localización geográfica

La distancia entre nodos vecinos se estima a través de la potencia de la señal recibida. También pueden conseguirse coordenadas relativas mediante el intercambio de dicha información. Finalmente, la localización de los nodos puede llevarse a cabo directamente mediante un sistema GPS.

- *Encaminamiento de minima energía basado en información geográfica* [29]. En GAF (*Geographic Adaptive Fidelity*) se divide la red en zonas formando una cuadrícula virtual. Cada nodo usa GPS para

asociar su posición a la malla. Para aumentar el ahorro de energía, algunos nodos de cada zona son desconectados en diferentes momentos. Por su parte, GEAR (*Geographic and Energy Aware Routing*) [30] selecciona los vecinos a los que enviar un paquete en términos de consumo energético y posición geográfica. Para enviar un paquete, un nodo observa si tiene algún vecino que esté más cerca de la región destino que él. En los envíos dentro de la su región, se usa tanto reenvío geográfico como inundación restringida si la densidad de nodos es baja.

Por forma de operación

- *Multicamino*: se estudian los protocolos que utilizan diversidad de caminos para llegar a la estación base. La tolerancia a fallos se mide en términos de la posibilidad de encontrar un camino alternativo entre fuente y destino cuando el camino primario falla. Esta característica puede mejorarse a costa de un mayor consumo de energía y generación de tráfico.

  - *Maximización de la vida de la red* [31]. Se propone un algoritmo que encamina los datos por la ruta que incluya los nodos tengan la mayor energía restante. Este camino será modificado dinámicamente según las condiciones de la red. Los costes de conmutación de los caminos no son cuantificados en el artículo donde se describe el funcionamiento del protocolo.

  - *Compromiso entre tráfico de control y fiabilidad multicamino* [32]. Se utiliza el multicamino para mejorar la fiabilidad de la red. Sin embargo, esta técnica implica un aumento del

tráfico por lo que el artículo estudia el compromiso entre ambos factores. La idea es dividir la información y enviarla por distintos caminos para poder reconstruir el original ante la pérdida de algunas de sus partes.

Por petición

El nodo destino envía una petición de datos a un nodo de la red. El protocolo de Difusión Directa [19] es un ejemplo de este tipo de operación. Por su parte, el Encaminamiento por Rumor [20] también utiliza este tipo de encaminamiento en el que los agentes buscan caminos más cortos o eficientes para llegar a los destinos.

Por negociación

Usan descriptores de datos de alto nivel para eliminar transmisiones redundantes. La familia de protocolos SPIN [18] es una muestra de este tipo de protocolos de encaminamiento.

Por calidad de servicio

La red debe buscar el compromiso entre la calidad de los datos y la energía consumida.

- *Encaminamiento por asignación secuencial* [33]. SAR (*Sequential Assignment Routing*) toma las decisiones de encaminamiento con base en tres factores: recursos energéticos, QoS en cada camino y nivel de prioridad del paquete. Se construyen árboles multicamino que engloban únicamente los nodos que cumplen los requisitos de energía y QoS requeridos.

- *SPEED* [34]. Requiere que cada nodo guarde información sobre sus vecinos y usa el reenvío geográfico para encontrar los caminos de encaminamiento. Provee eliminación de congestión.

Procesado coherente y no coherente

En general, los nodos colaboran entre sí para procesar diferentes datos de la red. El encaminamiento de procesado de datos no-coherente implica un procesado local de los datos antes de enviarlos a otros nodos para un procesado ulterior; estos nodos se conocen como *agregadores*.

En el encaminamiento coherente, los datos son enviados a los agregadores después de un procesado mínimo, que típicamente incluye tareas como señalización de tiempo, supresión de duplicados, etc.

- *SWE (Single Winner Algorithm) y MWE (Multiple Winner Algorithm)* [33]. En SWE un solo agregador es elegido para el procesado complejo. La elección se lleva a cabo por su reserva de energía y sus capacidades computacionales. Después del proceso SWE se habrá completado un árbol de saltos mínimos que cubrirá la red. Por su parte, MWE reduce el consumo de energía limitando el número de fuentes que pueden enviar datos directamente al agregador central. Se registra no sólo el mejor candidato para agregador sino los *n* mejores para cada nodo. Así, cada nodo tiene *n* caminos de mínima energía. Por último, se usa el SWE para finalizar el procesado.

1.4.- Escenario específico

Una vez estudiados las RAHI y RSI genéricas y distintos protocolos de encaminamiento definidos para las mismas, se presenta la descripción del escenario

específico donde se desarrollará el trabajo del presente PFM. Esta descripción atiende a la necesidad de analizar las diferentes posibilidades y escenarios que se presentan para la definición de la estructura y funcionamiento del algoritmo, antes de iniciar su diseño. Estas distintas opciones vendrán determinadas por una serie de factores que definirán unos escenarios de partida y que habrán de ser evaluados de forma que permitan la elección de aquélla que más se ajuste a las necesidades definidas previamente en el escenario elegido para la RSI. Así, se deberá tener en cuenta en el diseño las distintas alternativas y restricciones que se presentan en cuanto a:

- Estructura general y restricciones de la RSI.
- Características y condiciones particulares de los dispositivos inalámbricos sensores a utilizar: memoria, capacidad de procesado, etc.

1.4.1.- Estructura de la RSI real disponible

El presente proyecto surge de una necesidad concreta y su objetivo es el diseño de un protocolo que más tarde pueda implementarse en RSI reales y, específicamente, en las que utilizan dispositivos de la plataforma MICA como es el caso de las presentes actualmente en el Centro de Experimentación y Comunicaciones Inalámbricas (CECI), perteneciente al Departamento de Teoría de la Señal y Comunicaciones de la Universidad Rey Juan Carlos (URJC).

En la red de localización en interiores sobre tecnología ZigBee, en operación en la actualidad, es necesaria la implementación de un algoritmo de encaminamiento que sea capaz de enviar paquetes desde la pasarela central hacia un nodo sensor o desde un nodo sensor a otro cualquiera de la red sin necesidad de utilizar paquetes de control inundando la red cada vez que se pretenda realizar dicho envío. Hay varias razones por las que es conveniente buscar una alternativa a la inundación de la red: en primer lugar, para que circulen menos paquetes de control por la red, lo cual tiene innumerables ventajas y, en segundo, porque un

paquete destinado a un nodo en particular no sea procesado por todos los nodos de la misma. La reducción del número de paquetes que circulan por la red implicará una menor posibilidad de congestión y un ahorro de energía adicional significativo debido a que los nodos recibirán y enviarán menos paquetes. Ya se ha insistido en la importancia que el ahorro energético tiene en una RSI. En segundo término, si se añade cierta *inteligencia* a la red será posible el envío de paquetes únicamente allí donde se desea y por el camino indicado, sin que los demás nodos perciban el envío, lo cual redunda en ventajas adicionales además del ahorro energético de los nodos y de la disminución de la congestión. Estas *comunicaciones dirigidas* permiten diferenciar a los distintos nodos de la red entre sí e intercambiarse individualmente paquetes e información con cada uno de ellos, además de aumentar la seguridad de la RSI debido a que existen menos posibilidades de que un nodo espía que se haya introducido en la red escuche una comunicación particular entre dos nodos de la misma. Un conocimiento previo de diferentes rutas de envío de paquetes permite seleccionar diferentes alternativas en los caminos seguidos, lo cual redunda en un aumento considerable de la eficiencia del protocolo atendiendo a los paquetes recibidos con respecto a los que se pierden por el camino. Además, el retardo medio de los paquetes extremo a extremo también se vería reducido.

1.4.1.1- Protocolos de encaminamiento en sensores inalámbricos del tipo MICA (MICA, MICA2, MICAz)

El elemento sensor correspondiente a la plataforma MICA es el utilizado en el CECI, si bien lo que se indica a continuación se puede generalizar para otros sensores inalámbricos presentes en el mercado. Sus características y modos de funcionamiento se adaptan muy bien a aplicaciones en las que se establezca una estructura jerárquica entre los nodos (que forman las hojas del árbol) y el *Root* (raíz del árbol o pasarela).

Con respecto a estos dispositivos, el estudio de los protocolos de envío de paquetes implementados por el fabricante para estos sensores arroja inmediatamente como resultado que la comunicación entre estos dispositivos en excesivamente simple y primitiva, lo cual limita en gran medida la aplicación de estas redes en el mundo real.

Cuando una RSI se despliega en un entorno real con este tipo de dispositivos, únicamente existe una forma definida en que los sensores inalámbricos pueden enviar la información recogida a la pasarela: el protocolo *Collection Tree.* Este protocolo proporciona una forma de envío multisalto de paquetes en la red desde los nodos (hojas) hasta el elemento que actúa como colector de datos y al que nos referimos como *pasarela, estación base* o *Root*, aunque de ningún modo garantiza su recepción. Este envío se hace por contienda con el método *best-effort* y se basa fundamentalmente en el envío del paquete hacia el nodo que esté más cercano al *Root* que el que lo envía, para lo cual se utiliza el parámetro ETX (*expected transmissions*), el cual indica de forma numérica el nivel o el orden que tiene un nodo con respecto al Root (nivel=0). De esta forma, cuando un nodo quiere transmitir un mensaje, calcula este parámetro para todos sus vecinos y finalmente envía dicho paquete al que tenga menor valor de ETX.

Además, en cuanto al envío de paquetes entre el *Root* y los diferentes nodos de la red, únicamente se ha implementado el protocolo llamado *Dissemination* para el envío y control de versiones de software y programación en todos los nodos, su reconfiguración y el cambio del valor de una variable en todos ellos. En ningún caso los nodos guardan información acerca de otros nodos que tienen un nivel mayor o que están por debajo de ellos en la estructura jerárquica, es decir, de sus hijos.

1.4.1.2- Estructura de la red

Dicho lo anterior, el presente trabajo parte de una serie de sensores inalámbricos descritos anteriormente que, una vez desplegados, tratarán de formar una estructura jerárquica en el que el elemento raíz o *Root* será el encargado de proporcionar una interfaz con las demás redes exteriores, en las cuales, bien se recogerá la información generada por los sensores inalámbricos, o bien generarán datos que deberán ser enviados a un nodo particular de la red de sensores.

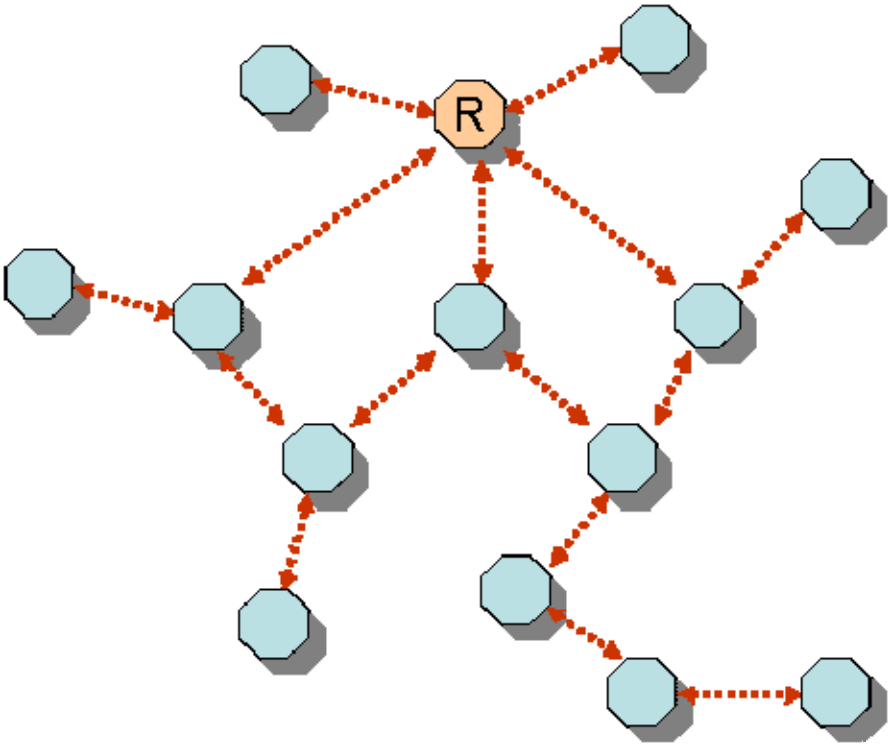


Figura 2. Estructura de una RSI

1.4.2.- Características particulares de los sensores inalámbricos

Los dispositivos electrónicos existentes hoy en día en el mercado son distintos y variados, pero con particularidades comunes. Fundamentalmente, estos

dispositivos poseen varias limitaciones que marcan sobremanera su utilización en proyectos reales, como son

- Deben ser baratos y de pequeño tamaño.
- Tienen una capacidad de almacenamiento y procesado de datos y de envío de información muy limitada.
- Deben maximizar la duración de sus baterías.
- Sus áreas de cobertura son muy limitadas ya que también lo está la potencia con la que emiten.

Como ya se ha afirmado previamente, las limitaciones anteriores han de tenerse en cuenta a la hora de diseñar el protocolo ya que marcarán su funcionamiento estableciendo ciertos límites, principalmente en cuanto a la capacidad de procesado y a la memoria total disponible para almacenar la información relativa a las diferentes rutas hacia otros nodos.

Para comprender mejor este tipo de dispositivos se incluye la siguiente figura, en la que se muestran las partes fundamentales de estos sensores inalámbricos:

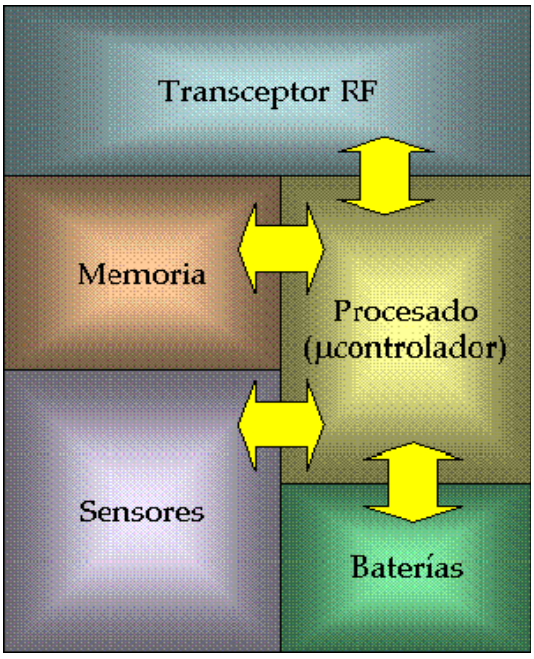


Figura 3. Estructura de un sensor inalámbrico

1. ***Subsistema de Procesado***: con el microcontrolador cono elemento principal, este subsistema se ocupa de la ejecución de programas y aplicaciones, de la gestión de recursos (memoria, baterías, etc.) y de la interacción con los distintos periféricos y la placa de sensado.
2. ***Subsistema de sensado***: incorpora los distintos sensores en la placa correspondiente: luz, humedad, temperatura, aceleración, sonido, etc.
3. ***Subsistema de memoria***: se compone de una memoria principal en la cual se almacenan los datos necesarios para la ejecución de los distintos algoritmos y otra memoria secundaria para el almacenamiento de los datos físicos recogidos del entorno en las operaciones de sensado. Esta memoria secundaria puede también almacenar temporalmente las imágenes o distintas versiones de programas o firmware que se distribuyen a través de la red.
4. ***Baterías***: regulan la tensión que proporcionan a los demás subsistemas.
5. ***Transceptor RF***: es el encargado de la transmisión y la recepción a través del medio inalámbrico de los paquetes de comunicaciones entre los distintos nodos de la red. Incluye todos los aspectos necesarios para el envío de la información: modulación/demodulación, codificación, etc.

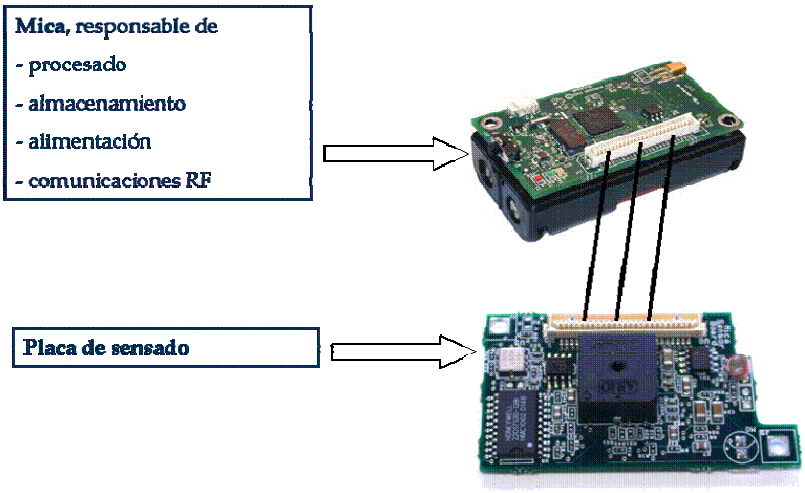


Figura 4. Dispositivo MICA de Crossbow

En cuanto a las características técnicas, en el *Apartado 7 Apéndices* se incluye la hoja de características del elemento sensor inalámbrico MICAz, si bien se adelantan algunas consideraciones técnicas de estos dispositivos:

- Poseen una memoria principal muy reducida, con valores de RAM de 4KB y una *Program Flash Memory* de 128KB.
- La capacidad de procesado del microcontrolador es limitada ya que trabaja a velocidades de unos 8MHz y suele tener una ALU de 8 bits. Este procesador incorpora una memoria EEPROM de programa de 4 KB .
- La tasa máxima de envío de paquetes de estos elementos es también muy reducida, con valores que rondan los 19.2Kbps (a 433 ó 916MHz).
- Su memoria Flash secundaria para almacenar los datos sensados es algo mayor que la primaria, básicamente con valores de 512KB.
- Su rango de cobertura puede alcanzar los 100 metros en exteriores y alrededor de 30 en interiores.
- Funcionan con dos pilas estándar tipo AA.

Para finalizar este apartado se incluye una figura que muestra las características técnicas de algunos de los dispositivos sensores inalámbricos más ampliamente utilizados en el mercado. Para consultar una lista más amplia de dispositivos puede visitarse la siguiente página web: http://www.btnode.ethz.ch/Projects/SensorNetworkMuseum

Mote Type	*WeC*	*René*	*René 2*	*Dot*	*Mica*	*Mica2Dot*	*Mica 2*	*Telos*
Year	1998	1999	2000	2000	2001	2002	2002	2004
Microcontroller								
Type	AT90LS8535		ATmega163		ATmega128			TI MSP430
Program memory (KB)	8		16		128			48
RAM (KB)	0.5		1		4			10
Active Power (mW)	15		15		15		60	0.5
Sleep Power (W)	45		45		75		75	2
Wakeup Time (s)	1000		36		180		180	6
Nonvolatile storage								
Chip	24LC256				AT45DB041B			ST M24M01S
Connection type	I^2C				SPI			I^2C
Size (KB)	32				512			128
Communication								
Radio	TR1000				TR1000	CC1000		CC2420
Data rate (kbps)	10				40	38.4		250
Modulation type	OOK				ASK	FSK		O-QPSK
Receive Power (mW)	9				12	29		38
Transmit Power at 0dBm (mW)	36				36	42		35
Power Consumption								
Minimum Operation (V)	2.7		2.7		2.7			1.8
Total Active Power (mW)	24				27	44	89	38.5
Programming and Sensor Interface								
Expansion	none	51-pin	51-pin	none	51-pin	19-pin	51-pin	10-pin
Communication	IEEE 1284 (programming) and RS232 (requires additional hardware)							USB
Integrated Sensors	no	no	no	yes	no	no	no	yes

Figura 5. Algunos de los sensores inalámbricos disponibles comercialmente.

2.- Objetivos

2.1.- Objetivo general y objetivos específicos

El objetivo principal del presente PFM es diseño de un protocolo de encaminamiento jerárquico para RSI. Este objetivo prioritario responde a las necesidades expresadas en el *Apartado 1 Introducción y antecedentes* en cuanto a la gestión y el control de las RSI, que implican una comunicación eficiente entre la estación base o pasarela y los nodos sensores, así como entre ellos mismos. Para conseguir la optimización de la eficiencia en dichas comunicaciones, se debe proveer un sistema de encaminamiento distinto del usado en la actualidad, basado en inundación, de tal forma que se maximice el aprovechamiento de recursos escasos como ancho de banda y energía.

La consecución final del objetivo principal del presente PFM se llevará a cabo a través de los siguientes objetivos particulares:

- Realizar un estudio del estado del arte en cuanto a los protocolos de encaminamiento en RSI

- Detectar lagunas de conocimiento o necesidades en algunos de los aspectos relacionados con el encaminamiento en RSI

- Analizar las diferentes alternativas de diseño de un protocolo jerárquico en RSI

- Construir el diseño teórico de un nuevo protocolo jerárquico de encaminamiento de paquetes de comunicaciones en una RSI que sustituya al método de inundación de la red con mensajes de control en comunicaciones pasarela-nodo y nodo-nodo

- Desarrollar una metodología de trabajo específica para el diseño del protocolo

- Proponer desarrollos, tareas y acciones futuras para la simulación e implementación posterior de este protocolo en redes reales

En el *Apartado 3 Metodología* se especifican más en detalle los objetivos concretos propuestos para cada una de las fases en que se ha dividido la realización de este proyecto.

2.2.- Requisitos de partida

La obtención de los resultados indicados anteriormente deberá atender a ciertas restricciones de partida tanto en lo referente a las condiciones de funcionamiento de las redes como a los requisitos impuestos a sus diferentes elementos, que condicionarán la definición del protocolo de encaminamiento jerárquico para RSI, objetivo principal del presente PFM. Obviamente no es lo mismo implementar un algoritmo que trabaje con nodos estáticos que hacerlo con funciones que manejen nodos que se mueven o nodos que aparecen y desaparecen en la red; como tampoco es lo mismo trabajar con altas tasas de envío de paquetes que con ratios bajos, en los que el tiempo que pasa entre dos envíos de paquetes en la red puede ser muy elevado.

Así, algunos de estos parámetros influirán decisivamente a la hora de diseñar las distintas funcionalidades del protocolo y otros (como la tasa de envío mencionada anteriormente) lo que harán será mejorar o empeorar la eficacia del protocolo diseñado en función de otros que trabajen por ejemplo con el método de inundación de la red.

Por tanto, las premisas básicas a tener en cuenta a la hora del diseño y definición de las funcionalidades de nuestro protocolo serán las siguientes:

- Existirá **un nodo que estará programado para actuar como pasarela o *Root*** en la red y cuyas características podrán ser diferentes a las del resto de nodos

- **Se permite la movilidad de todos los nodos de la red**, si bien la movilidad del *Root* afecta drásticamente al rendimiento de las transmisiones de la red por tratarse de un protocolo jerárquico

- El entorno o ambiente en el cual trabaja la red puede ser hostil, es decir, **pueden existir desvanecimientos temporales en los enlaces** entre nodos de forma que la comunicación entre dos de ellos se vea interrumpida durante un intervalo de tiempo determinado

- Un nodo podrá **asociarse con varios *nodos padre* simultáneamente**, de forma que pueda existir redundancia de rutas entre diferentes nodos

- Algunos nodos pueden dejar de funcionar correctamente y/o **apagarse temporal o definitivamente** debido al agotamiento de sus baterías, con lo que las rutas en las que ese nodo estuviese presente dejan de ser válidas para el encaminamiento de paquetes

- **Se permite la adición de nuevos nodos en la red** en posiciones aleatorias de la misma, siempre que el nodo introducido esté en la zona de cobertura de la menos un nodo ya presente en la red

3.- Metodología

El diseño de este nuevo protocolo jerárquico se ha desarrollado siguiendo una metodología de trabajo que facilite su desarrollo. Dicha metodología pretende, por un lado, evaluar y dimensionar el trabajo a desarrollar en función de los objetivos propuestos al inicio del proyecto y, por otro, dividir dicho proyecto en tareas más concretas que pudiesen ser ejecutadas en etapas sucesivas.

A través del correcto diseño de una metodología de trabajo se asegura el cumplimiento de las distintas tareas del proyecto y se hace hincapié en aquéllas más críticas a las que habrá que prestar especial atención y que pueden marcar el éxito o el fracaso del proyecto completo. Este desarrollo permite una mejor coordinación entre tutor y alumno, clarifica los objetivos, ayuda a dimensionar el problema y establece una relación directa entre las tareas a ejecutar y el tiempo en que se desarrollará cada una de ellas.

Además de delimitar las distintas fases del diseño de este nuevo protocolo jerárquico de encaminamiento, permite una planificación separada de cada una de ellas, es decir, la *modularización del proyecto* y, consecuentemente, favorece un reparto de tareas mucho más preciso entre el tutor y el alumno. Se trata, en suma, de aislar lo máximo posible los diversos componentes del diseño para poder manejarlos independientemente unos de otros con facilidad.

Para este PFM se ha adoptado una metodología similar a la que se hubiese podido desarrollar para la gestión de cualquier otro proyecto desarrollado en el ámbito profesional, teniendo en cuenta que para este trabajo dicha metodología se simplifica de forma considerable ya que muchos de los aspectos involucrados en la gestión de proyectos profesionales (proyectos en empresas privadas, proyectos europeos, etc.) no están presentes en un proyecto docente como éste. De esta forma, pueden considerarse en mayor o menor medida tareas propias de la gestión de proyectos como son la dirección del proyecto (llevada a cabo por el tutor), la planificación del mismo, la gestión de recursos (aunque únicamente el alumno sea el autor, deberá gestionarse su dedicación como recurso del

proyecto), el control de tareas y tiempos, la recopilación de datos e incluso podría hacerse una justificación o valoración económica del proyecto completo en función de sus resultados.

El uso de una metodología de trabajo permite además tener siempre presente el conjunto del proyecto y sus objetivos finales, pero también adentrarse en detalle en cada una de las tareas y sub-tareas definidas en cada etapa, de forma que pueda afirmarse que "*los árboles dejan ver el bosque*" o, dicho de forma más solemne "*Global think, local action*". La metodología empleada nos ha permitido adoptar una visión del proyecto denominada *TOP-DOWN*, es decir, analizar la información (tareas) por niveles de jerarquía, desde el conjunto hasta los detalles, nunca al revés.

La priorización de tareas es otro resultado de aplicar una metodología de proyectos a uno en concreto, si bien en el diseño que nos ocupa dicha priorización no es tan critica como en otro tipo de proyectos que sean menos lineales que éste (que podrían referirse como *matriciales*, en los que muchas de las tareas se solapan entre sí y se ejecutan simultáneamente en entornos diferentes y por personas distintas). En el caso de este PFM, la linealidad seguida en el desarrollo de las tareas es bastante clara por tratarse de tareas desarrolladas por una única persona (el alumno con la ayuda del tutor), si bien parte de las mismas bien pudiesen perfectamente haber estado parcialmente solapadas entre sí (por ejemplo, la codificación del algoritmo de construcción del árbol jerárquico se podría haber desarrollado inmediatamente después de su definición, mediante los diagramas de flujo, sin esperar a que estuviese completado el diseño del algoritmo final).

3.1.- Descripción de la metodología del PFM

La metodología de trabajo desarrollada para el presente PFM define las siguientes fases en el desarrollo del mismo:

- ***Fase 1: Definición del Proyecto***
- ***Fase 2: Planificación del Proyecto***
- ***Fase 3: Diseño Conceptual***
- ***Fase 4: Codificación y Programación Inicial***
- ***Fase 5: Elaboración de la Memoria Final del PFM***

A continuación de describen de manera detallada las actividades involucradas en cada una de ellas.

- **Fase 1: Definición del Proyecto**

El punto de partida para la elaboración de este PFM es la identificación de los objetivos a cumplir en el presente proyecto y, en consecuencia, la definición del alcance inicial y la profundidad requerida por este trabajo.

En el caso general, este alcance se establece una vez que se ha desarrollado un trabajo previo de búsqueda e identificación de lagunas de investigación, errores o maneras de optimizar determinados aspectos asociados a un tema de investigación.

Esta necesidad de un algoritmo, según lo desarrollado y expuesto en el *Apartado 1 Introducción*, es corroborada en el CECI, donde se da forma a la idea de desarrollar un nuevo protocolo de encaminamiento de paquetes de comunicaciones en una RSI ante la falta de un protocolo que sea capaz de comunicar la pasarela central de la red con cada una de los nodos sensores (también llamados *motas*) de manera individual o que permitiese a estas motas enviarse paquetes de unos a otros sin necesidad de inundar la red con mensajes de control. De ahí que se parta en este PFM de una necesidad detectada y a partir de ella se diseñen los siguientes pasos a seguir para la consecución del mismo.

Esta idea inicial, que de por sí establece ya unos objetivos concretos, debe ser confirmada posteriormente analizando en profundidad el estado del arte de los protocolos de encaminamiento en RAHI y en RSI y, más concretamente, el de los protocolos jerárquicos de comunicaciones.

Finalmente, una vez examinados los avances más significativos en el campo de los protocolos jerárquicos o no uniformes, esas metas se clarifican y concretan de forma explícita y, aunque pueden ser revisadas en etapas posteriores para adaptarlas a las circunstancias del proyecto, se pasa a la siguiente fase de identificación de las tareas y planificación del PFM.

En definitiva, esta primera fase puede dividirse en las siguientes tareas:

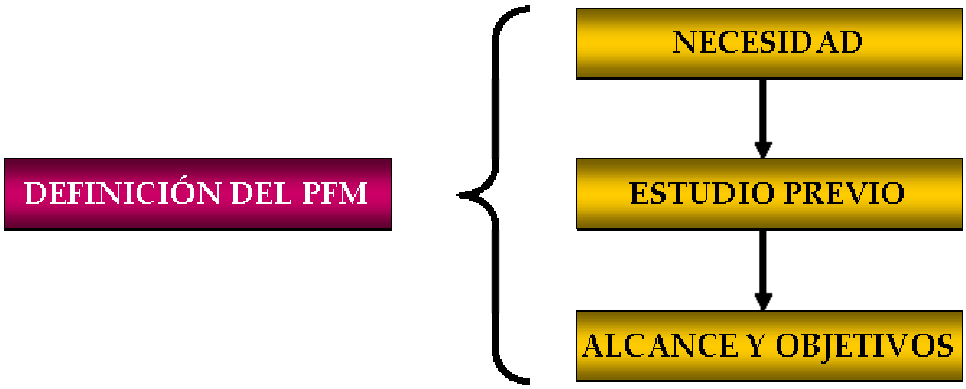


Figura 6. Esquema de la Fase 1 del PFM

- *Actividades principales:*
  - Identificación de las necesidades para la elaboración de un PFM.
  - Elaboración de un estudio previo de investigación para conocer el estado del arte en cuanto a los protocolos de encaminamiento en RAHI y RSI y, más en profundidad, en lo relativo a los protocolos jerárquicos.
  - Concreción de objetivos y metas y del alcance final del PFM.

- *Termina cuando:*
  - Exista una idea clara de la necesidad a cubrir por el proyecto.
  - Se haya desarrollado una labor de investigación que confirme la laguna de conocimiento en el tema particular del PFM.
  - Se propongan objetivos a cumplir en el desarrollo del proyecto.

- *Duración:*

  1 mes

- *Se solapa con:*

  Ninguna otra tarea.

- *Documentación que genera:*

  Apartados 1, 2 y 6 de este trabajo:

  - Introducción y Antecedentes: necesidad y descripción del proyecto
  - Objetivos a cumplir
  - Bibliografía utilizada

▪ Fase 2: Planificación del Proyecto

El propósito de esta fase es, por un lado, proporcionar una descripción detallada de las actividades o tareas a realizar dentro del ámbito de este PFM y, por otro, plasmar la planificación inicial del mismo relacionando las tareas entre sí y concretando un intervalo de realización de cada una de ellas. Esta fase puede ayudar a identificar aquellas tareas principales que van a requerir mayor esfuerzo y evaluar así los recursos necesarios para ponerlas en marcha y que puedan ser finalizadas de forma satisfactoria y en el plazo convenido. Una adecuada planificación del proyecto ayuda a garantizar que el avance se produzca de forma eficaz y que se establezca una base sólida para que el diseño final del protocolo sea exitoso.

Dentro de la etapa de planificación nos encontramos con diferentes tareas a realizar:

1. Resumen previo: en él se recogen inicialmente los aspectos más relevantes del PFM, tales como personas intervinientes y recursos, control del proyecto, objetivos propuestos, etc.

2. Estudio detallado: aunque el objetivo de este PFM sea el de diseñar un nuevo protocolo de encaminamiento para RSI, es necesario un estudio previo de los componentes y sistemas electrónicos presentes hoy en día en el mercado con el objetivo de que este protocolo pueda ser en un futuro implementado sobre sensores inalámbricos reales. Este aspecto es crítico y marcará drásticamente el desarrollo de las posibles soluciones o alternativas seleccionadas, ya que establecerá limitaciones en el diseño y en la cantidad de recursos disponibles para el correcto funcionamiento del protocolo una vez implementado.

Este estudio detallado debe finalizar con la presentación de las posibles alternativas de diseño del protocolo, conocidas también como *escenario,s* en los que debe desarrollarse nuestro diseño particular.

3. Definición de la estrategia de diseño: previamente al diseño del protocolo es fundamental identificar los elementos y circunstancias que deben ser tenidos en cuenta a la hora de diseñar el protocolo, así como los parámetros técnicos que influyen en el éxito o en el fracaso del protocolo una vez sea implementado. Se trata en definitiva de definir en qué aspectos centrar el diseño del protocolo para que mejore alguno de los parámetros que suelen utilizarse para medir la eficiencia de estos protocolos en RSI.

   Una vez definidos los parámetros de diseño, el paso siguiente es optar por una de las estrategias identificadas en el apartado anterior, que se corresponderá con aquélla que maximice la eficiencia de nuestro protocolo en función de algunos de los parámetros definidos.

   Puede ocurrir que esta selección no sea fácil y que se tenga que recurrir a algunos supuestos para poder llevarla a cabo.

4. Modularización del proyecto y definición de la secuencia de tareas a desarrollar: el siguiente paso es la división del proyecto global en tareas más concretas que puedan ser desarrolladas individualmente y el establecimiento de un orden de ejecución de las mismas.

5. Establecimiento del control del proyecto: aunque con mucha menor importancia que en un proyecto más amplio de investigación o uno empresarial, el control del estado del proyecto es fundamental para el desarrollo coordinado de las tareas definidas y su cumplimiento, lo que en definitiva marcará el éxito o el fracaso del mismo.

En nuestro caso serán necesarias reuniones periódicas de control del estado de las tareas que se estén desarrollando en ese momento concreto y para prevenir futuros problemas que puedan surgir en aquéllas que todavía no han comenzado a ejecutarse.

6. Asignación de recursos: uno de los últimos pasos de esta etapa es la asignación de recursos a las tareas encomendadas. Debido a que en un PFM los recursos humanos que lo desarrollan se limitan fundamentalmente al alumno y al tutor y se ahorra la gestión de otros recursos como pueden ser los económicos o los materiales, la implicación final que puede asociarse a esta etapa es la de dimensionar por separado cada tarea para poder posteriormente situar en un contexto temporal el proyecto global. En definitiva, se trata de ajustar las tareas a desarrollar a los requisitos demandados para un proyecto como éste.

7. Clarificación del alcance del PFM y revisión de metas y objetivos: una vez llegados a este punto, resulta conveniente la revisión de los objetivos iniciales y su ajuste en caso de que se haya producido la variación de alguna circunstancia inicial del proyecto.

Un resumen de las tareas incluidas en esta etapa es el siguiente:

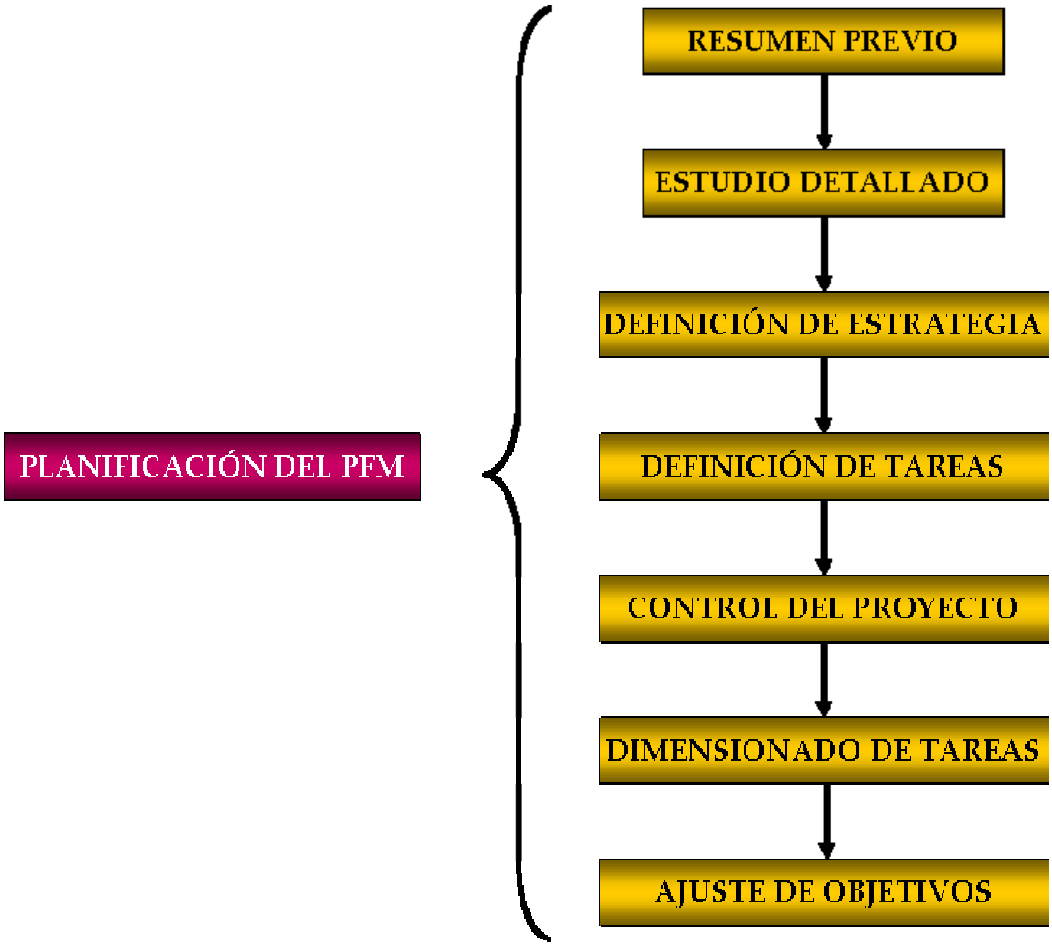


Figura 7. Esquema de la Fase 2 del PFM

- *Actividades principales:*
  - Estudio de las características y condiciones de funcionamiento de los diferentes tipos de sensores inalámbricos presentes en el mercado.
  - Definición de diferentes escenarios y alternativas de diseño del protocolo.
  - Definición de los parámetros de diseño en que se basará el nuevo protocolo jerárquico.
  - Selección de una alternativa entre todas las existentes para diseñar el protocolo.
  - División del proyecto en tareas concretas y establecimiento de un intervalo de ejecución de cada una de ellas hasta completar el proyecto global.
  - Definición del control del proyecto mediante la programación de reuniones periódicas.
  - Redefinición de los objetivos del proyecto si fuese necesario.

- *Termina cuando:*
  - Exista una idea clara de cómo desarrollar el proyecto.
  - Se hayan identificado todas las alternativas posibles y seleccionado una de ellas en función de alguno de los criterios manejados.
  - Se haya elaborado un cronograma de actividades identificando aquéllas más críticas o que pueden retrasar sensiblemente la finalización del PFM.

- *Duración:*

  2 meses

- *Se solapa con:*

  La parte inicial del estudio particular de los sensores inalámbricos puede solaparse con actuaciones de estudio similares realizadas en la fase anterior. El control del proyecto puede igualmente solaparse con la etapa anterior ya que no es una tarea independiente del resto de las de esta fase.

- *Documentación que entrega:*
  - Planificación temporal (Diagrama de Gantt) de las actividades a realizar.
  - Bibliografía utilizada.

- **Fase 3: Diseño Conceptual del Protocolo Jerárquico**

En esta fase se elaborarán los diferentes procesos y se desarrollarán los diagramas funcionales necesarios para comprender la estructura, características y funcionalidades de este nuevo protocolo jerárquico de encaminamiento.

Comenzando por los procesos principales que marcarán el funcionamiento básico y la estructura de los algoritmos que más tarde serán diseñados, se procederá a continuación con el diseño de aquellas funcionalidades que marcarán las características y la idiosincrasia propia del protocolo y lo diferenciarán de los existentes. Se hará especial hincapié en aquellos desarrollos y apartados que tienen en cuenta las características propias de los entornos inalámbricos y que los diferencian de las redes cableadas así como las propiedades de los sensores reales en los que se puede implementar.

En cuanto al método y al desarrollo del diseño, existe una multitud de maneras y formas de representar la información contenida y de plasmar los procesos y funcionalidades diseñadas de modo que pueda ser entendido por los demás lectores. Los más usuales son los siguientes:

- *Organigramas:* de diversos tipos (jerárquicos, de procesos, de aplicaciones, de proyectos). Principalmente se utilizan los organigramas funcionales y de proyecto para representar la estructura tanto organizativa como las funciones de cada uno de sus componentes.

- *Diagramas:* representaciones secuenciales que describen las tareas de un proceso, procedimiento o algoritmo mediante la utilización de símbolos predeterminados. Dentro de los diagramas, destacan los siguientes:

  - Diagrama de Flujo: describe el análisis y estructura de un proceso o procedimiento.

- Cuaderno de Carga: describe el contenido de una actividad, algoritmo o programa.

- Símbolos: identifican tareas y conceptos concretos, como por ejemplo una decisión, un cálculo, una llamada a una función o un resultado.

De los anteriores, el método seleccionado para la descripción y desarrollo de los diferentes procesos y procedimientos incluidos en el protocolo es el Diagrama de Flujo, también denominado f*lujograma*, muy adecuado por la forma en que se trata y visualiza la información contenida y ampliamente utilizado en el diseño de algoritmos computacionales.

Una parte fundamental de esta etapa es separar los distintos procesos que se van a desarrollar y agruparlos por funcionalidades, es decir, establecer una clasificación de los mismos en *procesos primarios* y *procesos secundarios*.

Los *procesos primarios* serán aquellos que definan la estructura principal del algoritmo, es decir, su funcionamiento básico y las llamadas a los demás procesos, a los que podemos referirnos como *procesos secundarios*, pero que tienen igual o mayor importancia ya que son los que establecen en gran medida la personalidad propia del protocolo, es decir, los que definen una manera propia de realizar las tareas y los que permiten dar robustez y flexibilidad al protocolo ante cambios en los parámetros físicos del entorno de la RSI en la que trabaja.

El siguiente paso, una vez separados los distintos procedimientos y procesos según su funcionalidad, consiste en el diseño en sí del protocolo. Además de tener en cuenta qué es lo que el protocolo debe hacer, es también una prioridad tener en mente los factores que han sido seleccionados en etapas previas y que permiten diseñar estos procesos optimizando la eficiencia del protocolo en cuanto a uno o varios de los parámetros de diseño elegidos.

Por último, tras un cuidadoso, preciso y no tan elemental ni efímero proceso de diseño, una vez finalizada la integración de todos los procesos y revisado por parte del alumno, el siguiente paso es el que se ha denominado *Control de Calidad de la fase de Diseño*. Esta última etapa de la fase 3 no es más que una revisión exhaustiva por parte del tutor de este trabajo y otras personas de una u otra forma relacionadas con el mundo de los protocolos de encaminamiento en redes inalámbricas, en la que se cuestionan y valoran los diferentes procesos y funciones incluidas en el diseño realizado. Evidentemente esta etapa tiene su parte de realimentación sobre el diseño, que acaba siendo modificado y optimizado todo lo que esta fase permite, para dotarlo de una estructura robusta y unas llamadas a procedimientos coherentes.

Para finalizar este apartado se ha de incidir en que, a pesar de que el objetivo principal de este diseño es que el protocolo funcione de la manera descrita y de forma correcta, es decir, sin errores, es muy difícil a pesar de la revisión exhaustiva poder garantizar a priori que los procesos diseñados estén libres de ellos o hayan sido optimizados en cuanto a rendimiento y eficacia del protocolo. Ciertamente esto no será así hasta que no se realice una implementación física del mismo, bien a través de una programación, que normalmente será orientada a objetos, y tras el paso de dicho protocolo por un simulador real, o bien mediante su programación directamente en los sensores inalámbricos reales utilizando el lenguaje y el sistema operativo de que disponen dichos elementos.

De la misma forma presentada en las fases anteriores, se expone a continuación un resumen de lo contenido en esta tercera etapa:

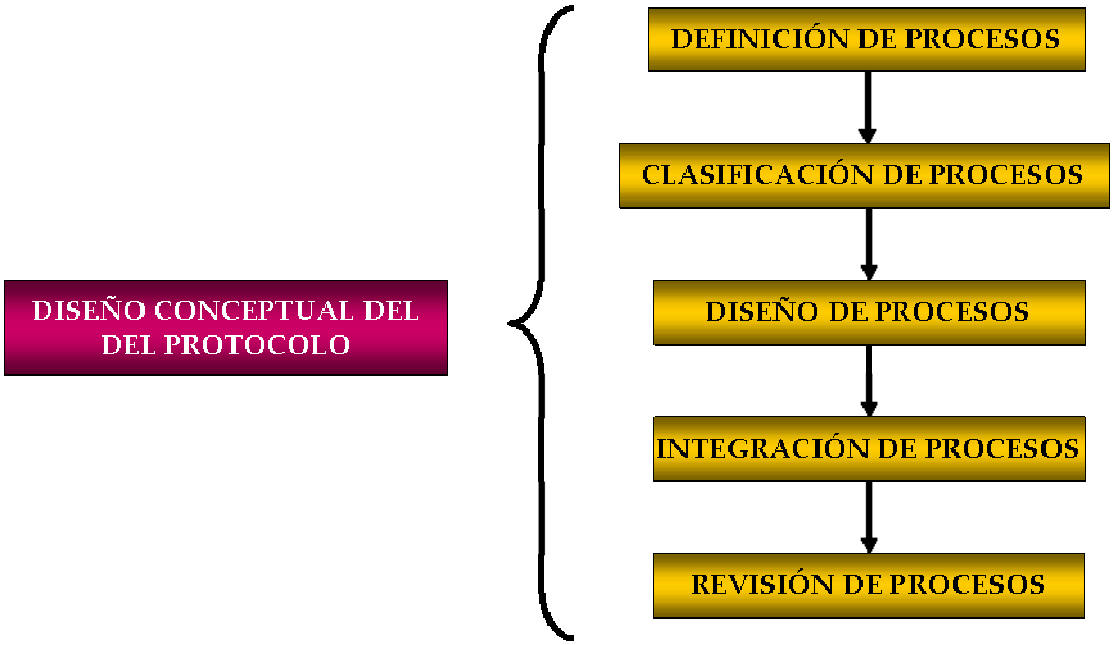


Figura 8. Esquema de la Fase 3 del PFM

- *Actividades principales:*
  - Definición de los distintos procesos y funciones en que se divide el diseño del protocolo, así como las particularidades y forma de trabajar de cada uno de ellos.
  - Clasificación de los procesos definidos atendiendo a la estructura y función principal del protocolo.
  - Establecimiento de llamadas e interfaces entre cada uno de los procesos definidos.
  - Diseño individual de todos los procesos del protocolo.
  - Integración y verificación de la coherencia y corrección de la estructura de llamadas a funciones y entre procesos.
  - Revisión del diseño de todos los procesos y redefinición u optimización de los mismos en los casos en que sea necesario.

- *Termina cuando:*
  - Se hayan diseñado todos los elementos necesarios para el correcto funcionamiento del protocolo jerárquico.

- Se hayan revisado dichos diseños y pueda garantizarse que existe una primera versión fiable y eficiente del mismo.
- Se esté en disposición de poder plasmar cada diagrama del diseño en una lista de acciones y llamadas a funciones mediante la utilización de elementos y sentencias propias de la programación a alto nivel: *if then*, *for do*, *while do*, etc. que sirva de interfaz entre el diseño teórico y una futura programación del protocolo en el lenguaje seleccionado.

❖ *Duración:*

2 meses

❖ *Se solapa con:*

Ninguna otra tarea.

❖ *Documentación que entrega:*

- Diagramas de Flujo de todos los procesos del diseño teórico.
- Documentación adicional que describe las particularidades de un proceso específico o las distintas posibilidades de diseño de alguno de ellos: diagramas temporales de envíos de mensajes, explicación detallada de una determinada función, etc.

- **Fase 4: Codificación de Procesos y Programación Inicial**

En este apartado se pretende plasmar la programación, a un nivel muy intuitivo, de alguno de los procesos diseñados pero utilizando la estructura e incluso los elementos propios de la programación de algoritmos de este tipo.

Esta programación disecciona cada proceso y lo divide en tareas más concretas, reduciéndolo a una sucesión de acciones y llamadas elementales, las cuales no son definidas en detalle sino que se establece qué hace cada una de ellas y cómo interactúa con el proceso completo.

Aunque pueda pensarse que esta programación inicial no es de utilidad, si se hace con precisión y se completa lo suficiente, puede llegar a contener toda la información necesaria para que un programador que esté familiarizado con este método no encuentre ninguna dificultad para programar el diseño de forma definitiva. Para ello, es necesario especificar claramente las tareas y acciones individuales, las variables utilizadas y sus valores, los parámetros a entregar a una función y el tipo de resultado que debe devolver, etc.

Este método de actuación es muy común en tareas de investigación y existen multitud de artículos publicados en revistas científicas que avalan esta forma de presentar los resultados. Las ventajas principales de esta programación básica e intuitiva pueden resumirse en las siguientes:

- Permite un mayor entendimiento de los procesos diseñados, ya que se concentra en los procesos en sí y en cómo interactúan unos con otros y describe sin entrar al detalle otras acciones más elementales.

- Permite especificar los parámetros y variables manejados en cada uno de los procesos, así como la definición de las llamadas entre ellos.

- Sirve de interfaz para una futura programación del algoritmo en algún lenguaje de los existentes en la actualidad, como por ejemplo en Java, o en C++.

- Es un método muy efectivo para corregir incorrecciones en el diseño y para detectar bucles o errores en las llamadas entre los diferentes procesos.

En consecuencia, se ha querido incluir en este PFM la programación de varios de los procesos diseñados y que conforman una de las partes principales del diseño, lo que se ha llamado el *algoritmo* o *rutina de construcción del árbol jerárquico*. Este algoritmo forma parte de la programación principal del sensor inalámbrico y su función es la de configurar una estructura jerárquica inicial cuando la RSI es desplegada, que permita poder iniciar las comunicaciones y el envío de paquetes entre dos nodos cualquiera de la red.

Para finalizar esta etapa, se muestra información acerca de las actividades incluidas y del material que se generaría tras su finalización:

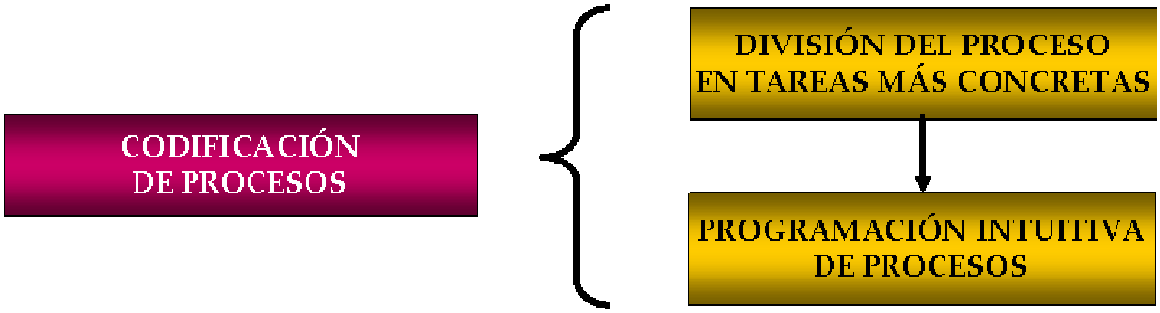


Figura 9. Esquema de la Fase 4 del PFM

❖ *Actividades principales:*

- ♦ Definición de los distintos procesos que incluye cada parte o rutina del diseño completo y que llegan a formar un algoritmo completo.
- ♦ División de cada proceso en acciones más elementales.
- ♦ Definición de variables y parámetros de intercambio en las llamadas entre procesos y funciones.
- ♦ Programación de los procesos seleccionados.

- Comprobación de la adecuación y coherencia de la programación con los diagramas de flujo utilizados para su desarrollo.

❖ *Termina cuando:*

- Se hayan programado en lenguaje pseudo-informático los procesos seleccionados.
- Se haya revisado la coherencia de esta programación con el diseño inicial.

❖ *Duración:*

2 meses

❖ *Se solapa con:*

Ninguna otra tarea. Es consecutiva a la fase 3, aunque también podría haberse optado por un solapamiento con la etapa 3 (codificación de los procesos inmediatamente después a su diseño).

❖ *Documentación que entrega:*

- Ficheros que contienen el código en lenguaje pseudo-informático de los procesos seleccionados.
- Documentación con el detalle de algún proceso.

▪ Fase 5: Elaboración de la Memoria Final del PFM

La última etapa del presente PFM comprende la elaboración de esta memoria y la inclusión de todos lo diagramas y códigos que han sido generados en el desarrollo del mismo.

❖ *Actividades principales:*

- ♦ Definición de la estructura de la memoria de este PFM.
- ♦ Redacción de los distintos apartados que la componen.
- ♦ Inclusión de diagramas de flujo, ficheros, código de procesos y cualquier tipo de información que haya sido utilizada para el diseño de este nuevo protocolo jerárquico.
- ♦ Revisión final por parte del tutor.

❖ *Termina cuando:*

Se revise, encuaderne y deposite la presente memoria.

❖ *Duración:*

1 mes

❖ *Se solapa con:*

Puede solaparse con cada fase finalizada documentando por escrito todo aquello que ha sido desarrollado en la etapa concluida.

❖ *Documentación que entrega:*

Memoria final del PFM encuadernada con toda la información asociada al diseño.

3.2.- Etapas del PFM y plan de trabajo

Una vez que se ha descrito la metodología propuesta para la realización de este PFM se está en condiciones de aplicarla al caso concreto que nos ocupa. El primer resultado de este proceso es la creación de un Diagrama de Gantt que refleje las distintas fases descritas anteriormente y las actividades a realizar en cada una de ellas. En definitiva, se trata de aplicar todo lo dicho anteriormente al caso del diseño de un protocolo jerárquico de encaminamiento para RSI.

Como resultado del desarrollo de las distintas fases se irá produciendo una serie de documentos, ficheros y diagramas de flujo y que, al integrarse unos con otros, formarán el *Apartado de Resultados* de este PFM ya que es esta documentación la que define el protocolo con sus funcionalidades.

En consecuencia, se presentará a continuación la metodología y las fases de trabajo que se han seguido en este PFM y se dejará para el *Apartado 4 Resultados* toda aquella documentación que refleje los logros alcanzados y que constituya de alguna u otra forma parte del diseño teórico del protocolo.

3.2.1.- Plan de trabajo

A continuación se muestra el plan de trabajo seguido para la confección de este PFM. En él pueden observarse las distintas etapas descritas a continuación y las características y particularidades propias de cada una de ellas, así como las tareas concretas a completar, los resultados de cada una y la duración final de todas ellas.

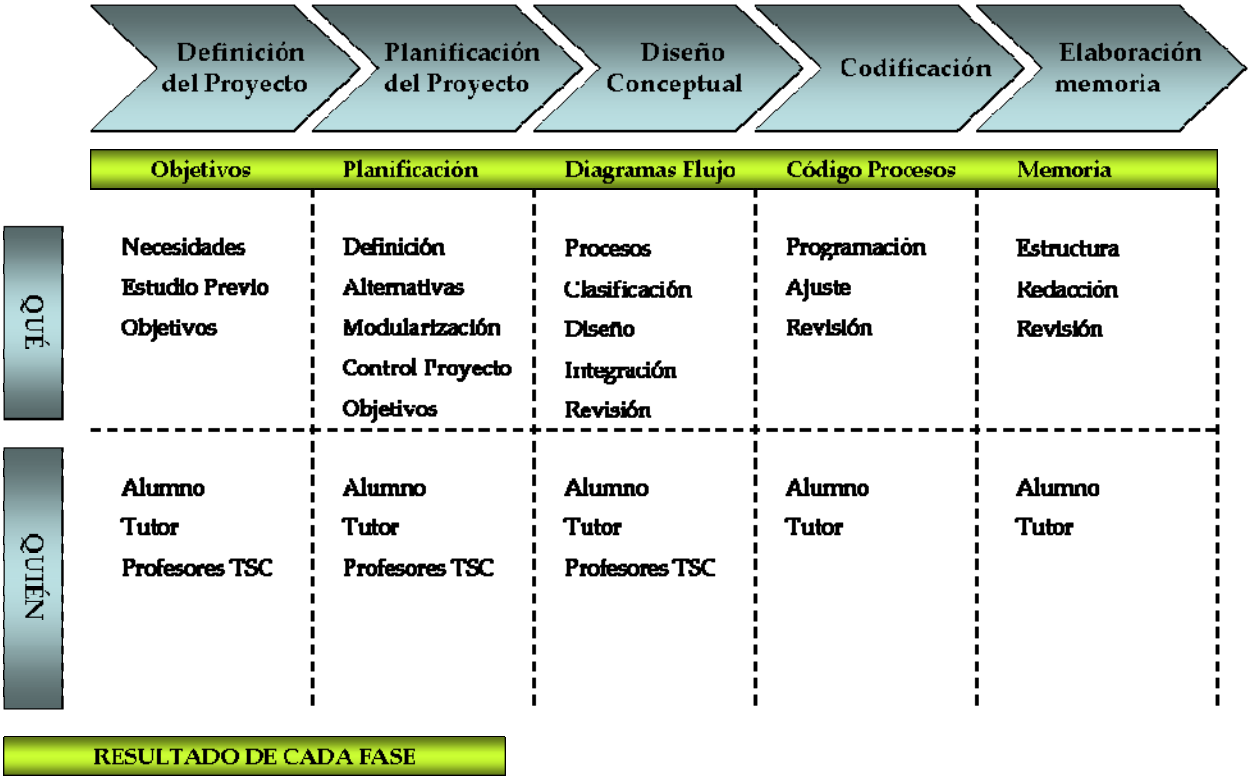


Figura 10. Plan de trabajo

Cada una de las etapas descritas se compone de las siguientes tareas concretas:

Paquete de trabajo	1.Definición del proyecto	Fecha Inicio	Mes 1	Fecha Fin	Mes 1
Objetivos Definir el tema y los objetivos del PFM					
Tareas **T1.1 Identificación de necesidades para la elaboración de un PFM**: sacar a la luz lagunas de conocimiento y definición de necesidades a cubrir en el encaminamiento en RSI y RAHI. **T1.2 Estudio del estado del arte de los protocolos de encaminamiento en RAHI y RSI**: revisión de protocolos existentes y comprobación de lagunas de conocimiento. **T1.3 Definición del tema del PFM**: **u**na vez identificadas las necesidades y revisado el estado actual, se define el tema de este PFM. **T1.4 Definición de objetivos iniciales en la elaboración del PFM**: elaboración de un listado de objetivos según los resultados de las tareas T1.1 y T1.2					
Entregables (mes en que tendrá lugar) **E1.1 Introducción y Necesidades (M1)**: documento que introduce el tema del encaminamiento en RSI y RAHI e identifica las necesidades concretas a cubrir. **E1.2 Objetivos (M1)**: documento con los objetivos iniciales del PFM.					

Paquete de trabajo	2.Planificación del proyecto	Fecha Inicio	Mes 2	Fecha Fin	Mes 3

Objetivos

Realizar una planificación detallada del PFM. Concretar las acciones y tareas a desarrollar y seleccionar las alternativas más adecuadas en cada caso atendiendo a la optimización de los parámetros de diseño. Establecer los criterios a seguir para el control del PFM.

Tareas

T2.1 Estudio de las características de los sensores inalámbricos reales: conocer las características y limitaciones de estos componentes en cuanto al almacenamiento, procesado y transmisión de información.

T2.2 Definición de distintas alternativas para el diseño de un protocolo jerárquico de encaminamiento en RAHI y RSI: definición de opciones técnicas para el diseño del protocolo.

T2.3 Definición los parámetros de diseño a tener en cuenta en el funcionamiento del protocolo: establecimiento de los parámetros que van a medir la eficiencia del protocolo y que son prioritarios en el diseño.

T2.4 Selección de un diseño técnico: escoger una alternativa entre todas las presentadas, que será aquella que optimice los parámetros de diseño definidos en la etapa anterior

T2.5 División del PFM en tareas y fases: modularización del proyecto y establecimiento de intervalos temporales de ejecución de cada una de las fases.

T2.6 Establecimiento de medidas para el control del proyecto: reuniones periódicas, revisión de las tareas, etc.

Entregables (mes en que tendrá lugar)

E2.1 Diagrama de Gantt del PFM (M3): documento que identifique las diferentes fases, las dimensione temporalmente y señale la relación temporal entre todas ellas.

E2.2 Análisis de alternativas (M3): documentos con las diferentes alternativas a seguir.

Paquete de trabajo	3. Diseño Conceptual	Fecha Inicio	Mes 4	Fecha Fin	Mes 5

Objetivos

Definición de los distintos procesos y funciones en los que se divide el protocolo. Diseño de los procesos identificados y establecimiento de relaciones y llamadas entre ellos. Integración y revisión del diseño.

Tareas

T3.1 Definición de los procesos que conforman el protocolo: dividir el diseño del protocolo en procesos o funcionalidades más concretas y clasificar cada uno de ellos en función de su alcance.

T3.2 Diseño de todos los procesos definidos: creación de un diagrama de flujo para cada proceso en el que se indique su funcionalidad y las acciones que ejecuta.

T3.3 Integración de los procesos: asegurar la coherencia entre las relaciones y llamadas entre los distintos procesos.

T3.4 Revisión de los procesos definidos: escoger una alternativa entre todas las presentadas, que será aquella que optimice los parámetros de diseño definidos en la etapa anterior.

Entregables (mes en que tendrá lugar)

E3.1 Diagramas de Flujo de todos los procesos definidos (M5): documento que describe la funcionalidad concreta de cada proceso definido y que conforma el diseño del protocolo jerárquico.

E3.2 Documentación adicional (M5): información que explica, aclara o documenta algún proceso, aspecto o función concreta del diseño.

Paquete de trabajo	4. Codificación de Procesos	Fecha Inicio	Mes 6	Fecha Fin	Mes 7

Objetivos

Realizar una programación básica, intuitiva e inicial de los procesos implementados que permita un mayor entendimiento de los procesos diseñados y facilite su optimización y programación definitiva.

Tareas

T4.1 Definición de los algoritmos necesarios para representar el comportamiento del protocolo: **d**efinición de los distintos procesos que incluye cada parte o rutina del diseño completo y que llegan a formar un algoritmo.

T4.2 Definición de variables y parámetros de la programación: identificación de las variables necesarias para la programación del protocolo.

T4.3 Programación de los procesos seleccionados: implementar el código completo de algunos procesos que completen una de las rutinas principales del protocolo.

T4.4 Verificación de la coherencia entre los diagramas de flujo y la programación implementada: análisis de las posibles incorrecciones generadas entre la fase de diseño de diagramas y la de implementación de la programación.

Entregables (mes en que tendrá lugar)

E4.1 Ficheros con el código de las rutinas seleccionadas (M7): documento que incorpora la programación a alto nivel de los procesos seleccionados

Paquete de trabajo	5. Elaboración de la memoria del PFM	Fecha Inicio	Mes 1	Fecha Fin	Mes 8

Objetivos

Elaborar la memoria del PFM incluyendo todos lo diagramas y códigos generados en el desarrollo del mismo.

Tareas

T5.1 Definición de la estructura de la memoria del PFM: identificar los requisitos incluidos en el *Reglamento PFM RESMovil* y creación de la estructura de la memoria.

T5.2 Redacción de la memoria: redactar los distintos apartados que la componen e incluir diagramas de flujo, gráficos y códigos generados en las distintas etapas del proyecto.

T5.3 Revisión final de la memoria del PFM: revisión final y correcciones de errores identificados por parte del tutor.

Entregables (mes en que tendrá lugar)

E5.1 Memoria encuadernada del PFM (M8): 1 copia en papel y otra en formato electrónico

Paquete Trabajo, Fases y Tareas del PFM	Inicio	Fin	Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6				Mes 7				Mes 8			
			1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. Definición del proyecto	1	1																														
T1.1 Identificación de necesidades para la elaboración de un PFM																																
T1.2 Estudio del estado del arte de los protocolos en RAHI y RSI																																
T1.3 Definición del tema del PFM																																
T1.4 Definición de objetivos iniciales en la elaboración del PFM																																
2. Planificación del proyecto	2	3																														
T2.1 Estudio de las características de los sensores inalámbricos reales																																
T2.2 Definición de distintas alternativas para el diseño de un protocolo jerárquico de encaminamiento en RAHI y RSI																																
T2.3 Definición de los parámetros de diseño a tener en cuenta en el funcionamiento del protocolo																																
T2.4 Selección de un diseño técnico																																
T2.5 División del PFM en tareas y fases																																
T2.6 Establecimiento de medidas para el control del proyecto																																
3. Diseño Conceptual	4	5																														
T3.1 Definición de los procesos que conforman el protocolo																																
T3.2 Diseño de todos los procesos definidos																																
T3.3 Integración de los procesos																																
T3.4 Revisión de los procesos definidos																																
4. Codificación de Procesos	6	7																														
T4.1 Definición de los algoritmos necesarios para representar el comportamiento del protocolo																																
T4.2 Definición de variables y parámetros de la programación																																
T4.3 Programación de los procesos seleccionados																																
T4.4 Verificación de la coherencia entre los diagramas de flujo y la programación implementada																																
5. Elaboración de la memoria del PFM	1	8																														
T5.1 Definición de la estructura de la memoria del PFM																																
T5.2 Redacción de la memoria																																
T5.3 Revisión final de la memoria del PFM																																

Figura 11. Diagrama de Gantt del PFM

4.- Resultados

En los siguientes apartados se incluyen todos los resultados producidos en el proceso del diseño de un nuevo protocolo jerárquico para encaminamiento de paquetes en RSI.

Además de incluir los Diagramas de Flujo que muestran el funcionamiento en detalle de los distintos algoritmos, funciones y rutinas incluidos, se dedica una parte inicial a la descripción de las diferentes alternativas a seguir en el diseño, y una explicación de en qué consistiría cada una, finalizando dicho apartado con la justificación de la elección de una de ellas.

Se incluye también en este apartado una descripción de los distintos módulos o partes principales de los que debería componerse el protocolo final atendiendo a las funcionalidades definidas para el mismo. Una vez hecha esta clasificación, la siguiente etapa es la división de todos los módulos en los distintos procesos que los componen, describiendo cada uno de ellos y analizando su funcionamiento correspondiente.

Este *Apartado 4 Resultados* incluye además una parte dedicada a la estructura y definición de los campos principales que deberían tener los mensajes de control implementados en el protocolo y que son intercambiados por los nodos de la red para un correcto funcionamiento del mismo.

Por último, como ya se ha adelantado en el *Apartado 3 Metodología*, se ha procedido a la programación y codificación inicial de uno de los módulos principales del protocolo con todos sus procesos y llamadas incluidas, de forma que se proporcione una herramienta más útil y visual para una futura programación en lenguajes mucho más específicos. El algoritmo implementado es el que sirve para construir y actualizar dinámicamente el árbol jerárquico, proceso fundamental y en el que se basa la estructura y el funcionamiento de este protocolo.

En resumen, las partes principales en que se divide el *Apartado 4 Resultados* son las siguientes:

- ***Alternativas de Diseño***
- ***Módulos principales del protocolo***
- ***Procesos que conforman el protocolo***
- ***Estructura de los Mensajes de Control y de la Tabla de Rutas***
- ***Programación del algoritmo de Construcción del Árbol Jerárquico***

4.1.- Alternativas de Diseño

Sobre la base de las condiciones expresadas en los *Apartado 1.4 Escenario específico* y *Apartado 2.2 Requisitos de partida*, se describen las dos alternativas que se plantean en cuanto al funcionamiento del protocolo jerárquico objeto del presente PFM.

4.1.1.- Protocolo jerárquico con encaminamiento salto a salto de paquetes de datos

Esta alternativa, al igual que la segunda planteada, se inicia con la construcción de un árbol jerárquico en el que cada elemento se asocia a otros nodos y se establecen distintos niveles en función de la lejanía o proximidad que los nodos tengan con el elemento raíz o *Root.* En definitiva, el proceso consiste en formar un árbol cuya cabeza la ocupa el nodo *Root* y del que salen diferentes ramas; estas ramas tienen hojas (hijas del *Root* pero padres de las siguientes) y de las cuales también pueden salir otras ramas con hojas (hijos), que a su vez también pueden generar más ramas y más hojas.

Se trata, en resumen, de formar una estructura con niveles como en la figura siguiente, en la que para cada nodo (excepto el *Root* R que tiene nivel 0) se representa su nivel en la red mediante un número en su interior:

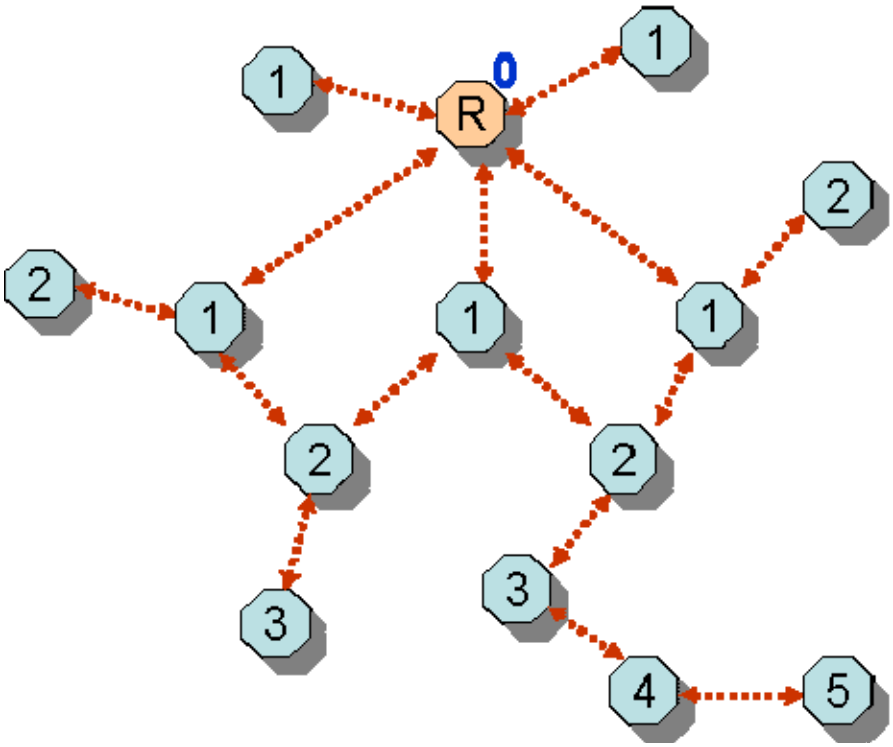


Figura 12. Construcción del árbol jerárquico

El proceso lo inicia el nodo *Root* enviando un paquete *broadcast* con su número de nivel (el 0). El nodo que reciba este paquete se asignará a sí mismo un nivel superior al del nodo recibido, es decir, un valor igual a uno. Una vez comunicado este nivel al nodo *Root*, éste lo aceptará como hijo y ambos guardarán estos datos en una tabla para futuras labores de encaminamiento.

Durante el proceso de formación del árbol jerárquico los demás nodos de la red se asignan un nivel en función del nivel del nodo superior, comunicándoselo posteriormente y actualizándose la tabla de encaminamiento en todos ellos con los datos de todos los nodos vecinos. De esta forma, cada nodo tendrá uno o varios nodos llamados *padre*, a los que enviará el paquete de datos si no tiene conocimiento del destino. También pueden tener uno o más nodos llamados *hijo*, es decir, que

están asociados con él y que tienen un nivel superior (en número) según lo explicado anteriormente.

Cuando un nodo intercambia información de nivel y actualiza información sobre algún nodo que tiene por debajo (es decir, ha asociado un nuevo hijo), automáticamente envía esa información a los nodos que están por encima de él en la jerarquía de la red (sus nodos padre) para que éstos tengan conocimiento de nuevas rutas hacia unos nodos de los que hasta ese momento desconocían incluso su existencia. Estos nodos reenviarán a su vez dicha información hasta llegar al nodo *Root*.

Como consecuencia de este proceso de reenvío de rutas de nodos pertenecientes a ramas inferiores del árbol, a medida que se sube hacia el nodo *Root* los nodos van almacenando cada vez mayor cantidad de rutas en sus tablas. Estas rutas son las correspondientes a todos los nodos que un nodo en particular tiene por debajo en alguna de sus ramas, por lo que si le llega un mensaje con destino a alguno de ellos automáticamente sabrá hacia dónde reenviarlo para que llegue correctamente a su destino.

En caso de que un nodo reciba un paquete y no tenga información en sus tablas de rutas hacia su destino, automáticamente lo reenviará a su nodo padre (y si tiene varios, a uno de ellos), el cual repetirá el proceso hasta que alguno de los que reciben el paquete tenga conocimiento de la menos una ruta hacia el destino. Si el paquete llega hasta el *Root* y éste tampoco tiene conocimiento del nodo destino, iniciará una búsqueda por inundación en la red para la detección del mismo.

Se puede decir que el encaminamiento de los paquetes de datos es el denominado *salto a salto*, es decir, cada salto del paquete (el nodo siguiente al que se envía) se decide únicamente en el nodo que lo envía y, a continuación, será el nodo que lo recibe el que decidirá el siguiente salto.

4.1.2.- Protocolo jerárquico con encaminamiento de paquetes de datos en origen a partir de ROOT

Esta alternativa tiene una parte común con la anterior en cuanto a la formación de niveles del árbol atendiendo a la proximidad al nodo *Root*. Sin embargo, existe una diferencia fundamental que marca el funcionamiento de esta otra versión del protocolo.

De forma similar al caso anterior, cuando un nodo se asocia a uno o varios nodos padre, a continuación almacena esos nodos en su tabla de rutas. Sin embargo, al contrario que la alternativa anterior, cuando un nodo asocia un nuevo hijo en este caso no actualiza su tabla de rutas con esa información, sino que automáticamente la envía a sus nodos padre. Ahora el nodo que recibe esta ruta tampoco la almacena en su tabla sino que directamente la vuelve a reenviar hacia sus nodos padres, y así sucesivamente hasta alcanzar el nodo *Root*. De esta forma, se producen dos situaciones significativas:

- Cada nodo de la red a excepción de *Root* sólo tiene almacenada información de sus nodos padre

- El nodo *Root* posee información de las rutas hacia cualquier nodo de la red

Como consecuencia de ello, cuando un nodo genera o recibe un paquete de datos hacia un destino que no es él, automáticamente reenvía el paquete a sus padres para que dicho paquete de datos alcance el nodo *Root* (lo que se producirá siempre que el nodo destino no se encuentre en el camino directo hasta el nodo raíz), que es el que tiene información de la ruta que deberá seguir el paquete hasta alcanzar el destino.

De esta forma, la mayoría de los paquetes de datos que circulan por la red (a excepción, como se ha mencionado anteriormente, de aquéllos en los que el destinatario del paquete se encuentre en el camino directo de subida desde el origen hasta el nodo raíz) pasarán por el nodo *Root*, el cual contiene información acerca de todos los nodos de la red y por tanto encaminará dichos mensajes insertando en la cabecera del propio paquete la secuencia de nodos que éste deberá seguir para alcanzar su destino. En estos casos, los paquetes seguirán dos trayectorias diferentes: una *de subida* hasta el nodo *Root* pasando por nodos intermedios y después otra *de bajada* hacia el nodo destino, pudiendo incluso a volver por nodos por los que ya ha pasado anteriormente en el camino de subida.

Esta alternativa se denomina en el presente PFM *con encaminamiento de paquetes de datos en origen a partir de Root* de acuerdo con el razonamiento expuesto en los párrafos anteriores.

Para finalizar, en realidad se puede decir que la expuesta anteriormente es una versión extrema del protocolo, en la cual ningún nodo excepto *Root* guarda ningún tipo de información en su tabla de rutas excepto la de sus nodos padre, y esto obligatoriamente es necesario para poder encaminar hacia ellos los paquetes de datos recibidos o generados en el nodo. Sin embargo, puede pensarse en opciones intermedias entre las dos versiones expuestas, es decir, versiones en las que un nodo almacene cierta información en sus tableas, principalmente la relativa a todos sus nodos vecinos: padres, hijos o vecinos del mismo nivel. En este caso se evitaría, por ejemplo, que un paquete circule por la red hasta el nodo raíz y vuelva al nodo origen cuando éste quiera enviar datos a uno de sus hijos o vecinos.

4.1.3.- Selección de alternativa principal de diseño

4.1.3.1- Parámetros de medida

A la hora de medir la eficiencia de un determinado protocolo de encaminamiento es necesario definir cómo dicha eficiencia va a ser medida, es decir, qué aspectos o parámetros son los que se van a medir y cuáles de ellos determinan si un protocolo es mejor o peor que otro en función de un determinado resultado.

Ya se ha mencionado anteriormente que una de las principales ventajas de utilizar un protocolo inteligente es que en principio está diseñado para el ahorro de paquetes de control, con lo que se puede partir de este punto para empezar a definir algunos de los parámetros a medir que nos indicarán la eficiencia de nuestro protocolo.

La lista de algunos de los parámetros que pueden llegar a definir la eficiencia de un protocolo de encaminamiento en una RSI es la siguiente:

parámetro	*qué mide*
Promedio del nº paquetes de Control con respecto al total (control + datos)	*La relación entre el nº de paquetes de control necesarios por cada paquete de datos enviado para el funcionamiento correcto del protocolo*
Retardo medio de un paquete extremo a extremo	*Tiempo medio que tarda un paquete de datos desde que se pone a disposición de un nodo para ser emitido hasta que es recibido correctamente por el destino*
Tiempo medio de adquisición de ruta	*Tiempo medio que se necesita para conocer una ruta hacia un nodo destino cuando ésta es solicitada*
Nº medio de retransmisiones de paquetes	*Cuántas veces promedio hay que enviar un mismo paquete para que llegue con éxito al destino*
Porcentaje medio de paquetes entregados con éxito con respecto a los tx	*Es una medida de la eficiencia del protocolo ya que refleja el porcentaje de paquetes que se transmiten y alcanzan el destino, es decir, que no se pierden en algún punto de la ruta*

Tasa de almacenamiento medio por nodo	*Representa el promedio de bytes de memoria que se requiere por nodo para el correcto funcionamiento del protocolo. En esta memoria se almacenarán las rutas y datos de padres, hijos, vecinos, etc. estipuladas en el funcionamiento del protocolo*
Tasa máxima de envío de datos	*Velocidad máxima de envío de paquetes de datos que no degrade la eficiencia del protocolo por debajo de un valor mínimo*
Longitud media del camino	*Longitud media del camino seguido por un paquete de datos desde el origen hasta el destino. Este parámetro es una manera de medir la directividad de un protocolo*
Promedio de vida de un nodo	*Duración media de las baterías y fuentes de alimentación de los nodos*
Escalabilidad de la red	*Grado en que se pueden añadir más nodos a la red o ésta se pueda ampliar con otras redes sin que el protocolo de encaminamiento disminuya su eficiencia manteniendo valores mínimos*

Tabla 1. Parámetros de medida de eficiencia de encaminamiento en RSI

No se ha de olvidar que todos estos parámetros deben ser evaluados teniendo en cuenta el contexto y las situaciones en las que están inmersos los nodos de la red, las cuales influyen de manera decisiva en el desempeño del protocolo. Existen por tanto otros parámetros que definen cómo son los patrones seguidos en el funcionamiento y en la simulación del protocolo y sirven para establecer contextos equivalentes a la hora de comparar el rendimiento de diferentes protocolos en una misma red.

Así, para evaluar la eficiencia de un protocolo hay que fijar previamente los siguientes patrones y variables:

1. El tamaño de la red.
2. Densidad de la red: número promedio de vecinos de un nodo.
3. Movilidad de los nodos: patrón que establece en qué momento, a qué velocidad y en qué dirección un nodo cambia su posición.
4. Tasa de cambio de la topología: medida de la velocidad con la cual la topología de la red está cambiando.

5. Capacidad del enlace: velocidad efectiva de un enlace entre dos nodos medida en bits/segundos, descontando pérdidas debidas a accesos múltiples, codificación, etc.
6. Modelos de tráfico de datos: patrón que establece cómo, con qué tasa y dónde se genera el tráfico en la red y que marca cómo debe ser la adaptación del protocolo a patrones de tráfico no uniformes o a ráfagas.
7. Fracción y frecuencia de nodos inactivos (*sleeping mode*): modelo que establece la actividad/inactividad de los nodos debido fundamentalmente a que entran en estado *dormido"* para ahorrar energía y que, por tanto, dejan de poder encaminar paquetes durante instantes de tiempo determinados.

4.1.3.2- Análisis de las alternativas

Atendiendo al funcionamiento descrito anteriormente para cada alternativa planteada y a los parámetros que van a medir la eficiencia del protocolo, pueden establecerse las siguientes conclusiones:

- Atendiendo al *número de paquetes de control necesarios con respecto al total*, ambas alternativas no parecen presentar diferencias sustanciales, si bien la alternativa de encaminamiento salto a salto generalmente suele necesitar menos paquetes de control en el reenvío de rutas hacia los nodos superiores ya que los suprime si esta información es redundante.
- Considerando el *retardo medio extremo a extremo* es de suponer que las rutas seguidas en la variante de encaminamiento en origen a partir del *Root* serán sensiblemente más largas que las del encaminamiento salto a salto ya que casi todas ellas pasarán por el nodo *Root* antes de llegar al destino (camino *de subida* + camino *de bajada*), mientras que en el segundo caso sólo lo harán si el destino

se encuentra en una rama del nodo *Root* diferente de la del nodo origen o se desconoce dicha posición.

- En cuanto al *tiempo medio de adquisición de ruta* y *la longitud media del camino,* el razonamiento es similar al indicado en el punto anterior relativo al retardo extremo a extremo, con lo que el encaminamiento salto a salto parece comportarse mejor que el de origen.
- Con respecto al *número medio de retransmisiones de paquetes* y *el porcentaje medio de paquetes entregados con éxito*, no existen a priori diferencias apreciables entre las dos alternativas.
- La alternativa de encaminamiento en origen desde el *Root* optimiza sin embargo el parámetro que indica la *tasa de almacenamiento medio por nodo,* ya que el único dato relativo a las rutas que guardan los nodos (excepto *Root*) es el de sus nodos padre.
- En relación al parámetro que mide la *tasa máxima de envío de datos* permitida para no degradar la eficiencia del protocolo, es de suponer que será peor cuantos más paquetes circulen simultáneamente por la red, que tiene mucho que ver con el retardo extremo a extremo y la longitud del camino recorrido, por lo que puede intuirse que el encaminamiento salto a salto pueda mejorar estos valores con respecto a la variante en origen desde *Root.*
- Teniendo en cuenta el *tiempo promedio de vida de un nodo,* puede preverse que, si bien las diferencias entre ambas alternativas pueden no ser significativas, la alternativa de encaminamiento en origen desde *Root* optimiza el uso del controlador al utilizar un número menor de operaciones para el procesado y actualización de las rutas en memoria.
- Por último, en relación a la *escalabilidad de la red* puede realizarse el mismo razonamiento que el relativo al primero de los parámetros

objeto de estudio, el *número de paquetes de control necesarios con respecto al total*, ya que son dos parámetros que están relacionados directamente entre sí.

Teniendo en cuenta los razonamientos realizados en los puntos anteriores, la opción seleccionada para el diseño del protocolo jerárquico es la de **encaminamiento salto a salto**. Un posterior estudio detallado, simulación e implementación del mismo arrojarán resultados veraces acerca de la eficiencia del protocolo atendiendo a los parámetros anteriormente citados.

4.2.- Módulos principales del protocolo

Una vez seleccionada la alternativa a seguir, el siguiente paso será el de dividir el algoritmo final en módulos funcionales concretos que se ocupen de las distintas funcionalidades definidas en el protocolo.

En consecuencia, se han definido tres módulos principales que conforman el funcionamiento completo del protocolo jerárquico:

1. **Construcción y actualización dinámica del árbol jerárquico**
2. **Envío de un paquete de datos desde un nodo cualquiera de la red**
3. **Recuperación de la red frente a errores y cambios en su estructura**

4.2.1.- Construcción y actualización dinámica del árbol jerárquico

Esta funcionalidad ha sido ya introducida en el apartado anterior y su misión es la de conformar inicialmente una estructura de padres e hijos que proporcione a cada nodo de la red una posición y un nivel determinados dentro de

la misma que le permitan el envío y la recepción de paquetes de datos a otros nodos de la red.

Además de para construir inicialmente el árbol jerárquico, este módulo incorpora funciones que permiten la inclusión en la estructura de la red de nuevos nodos así como la actualización dinámica de dicha estructura en aquellos casos en los cuales se haya producido alguna variación con respecto a la formación inicial.

Los procesos y funciones incluidos en este módulo son:

- Envío del paquete de control CHREQ (*"**CH**ild **REQ**uest"*)
- Envío del paquete de control CHREP (*"**CH**ild **REP**ly"*)
- Envío del paquete de control CHACCEPT (*"**CH**ild **ACCEPT**"*)
- Selección de nodo padre tras recibir varios CHREQ en un intervalo
- Acciones ejecutadas sobre la tabla de rutas: buscar, añadir o eliminar un registro
- Envío del paquete de control CHROUTE (*"**CH**ild **ROUTE**"*)

4.2.2.- Envío de un paquete de datos desde un nodo de la red

Este módulo implementa las acciones a tomar por parte de los nodos tras la formación del árbol jerárquico cuando pretendan enviar un paquete de datos con destino a otro nodo de la red.

Para el diseño de este módulo se han evaluado distintas opciones, atendiendo a si el envío de un paquete de datos se realiza a través de todas las rutas posibles o únicamente a través de una de ellas. Esta posibilidad de envío múltiple surge al permitirse durante la construcción del árbol la asociación de un nodo a varios padres y, en consecuencia, la redundancia de rutas entre nodos.

El presente protocolo se ha diseñado partiendo de la premisa de que un nodo realiza el envío del paquete únicamente a través de una de las rutas, la cual selecciona previamente atendiendo a parámetros de calidad de los enlaces, es decir, no existirá duplicidad de rutas en los paquetes enviados. Sin embargo, puede ocurrir que la ruta seleccionada no sea válida por alguna razón concreta y el paquete enviado no llegue a su destino. En este caso, el nodo escogerá una segunda ruta para su envío, y así sucesivamente hasta que el paquete se entregue con éxito o se agoten todas las posibilidades en cuanto a caminos distintos a seguir.

Para el correcto funcionamiento de este módulo se han desarrollado los siguientes procesos y funciones:

- Selección de ruta óptima
- Iniciar búsqueda de un nodo por inundación (*Root*)
- Selección de nodo padre
- Búsqueda de ruta alternativa

4.2.3.- Recuperación de la red frente a errores y cambios en su estructura

La recuperación de la red frente a errores, pérdidas de enlaces y cambios en la estructura de la red o en la posición de los nodos dentro de la misma es sumamente importante para optimizar el rendimiento del protocolo jerárquico.

Los fallos o nuevas situaciones que pueden producirse y que afectan a la estructura del árbol jerárquico construido son principalmente los siguientes:

- Pérdida temporal de un enlace entre nodos debido a desvanecimientos, interferencias, ruido, etc.
- Movimiento de nodos que hace variar la estructura del árbol jerárquico

- Aparición de nuevos nodos en la red
- Desaparición o apagado temporal de nodos de la red

De ellos, el movimiento de un nodo en la red puede tratarse como una desaparición + aparición en otro lugar. Por tanto, los casos a estudiar se simplifican y quedan de la siguiente forma:

1) **Desaparición de un nodo de la red**
   a) El nodo no tiene hijos
   b) El nodo tiene hijos
2) **Pérdida de un enlace entre nodos**
3) **Aparición de un nodo nuevo en la red**

1.a) Actualización de rutas por desaparición de nodo sin hijos en la red

Se supone que el nodo que desaparece tiene al menos un nodo padre asociado.

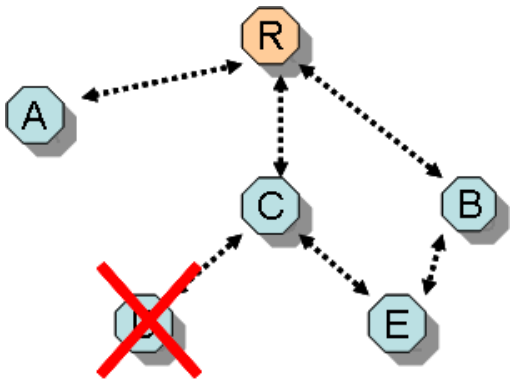


Figura 13. Desaparición de hijos

OPCIÓN 1: Para detectar la pérdida de algún vecino es necesario mantener actualizada una tabla con los vecinos actuales, lo que implica el envío periódico de mensajes de control entre ellos.

Cada cierto tiempo, *Root* envía mensaje HELLO con su identificador (ID). Sus vecinos e hijos lo reciben y responden de forma análoga con otro HELLO y su correspondiente ID. De esta forma el mensaje de control se va propagando a lo largo del árbol desde arriba hacia abajo y todos los nodos tienen conocimiento de los nodos vecinos (padres, hijos y nodos del mismo nivel jerárquico) que tienen en ese momento determinado.

Con esa información se consulta la tabla de vecinos y se detecta aquéllos que han desaparecido, bien porque el nodo ya no está en esa posición, o bien porque se ha perdido temporal o definitivamente el enlace con él. A continuación se envía un mensaje CHDEL (*"**CH**ild **DEL**ete"*) a los padres para que borren de sus tablas las rutas que contengan al vecino desaparecido.

Puede ocurrir que el vecino que desaparece se mueva cerca de un nodo de menor nivel, con lo que deberá asociarse con él como hijo (proceso que se contempla en el módulo principal de *Construcción y actualización dinámica del árbol jerárquico*). Si después de hacerlo, al nodo padre con el que se ha asociado le llega un mensaje CHDEL procedente del nodo que detectó la pérdida, deberá borrar las rutas antiguas en caso de tenerlas en sus tabla pero no la nueva, es decir, la recién creada con su nuevo hijo, que es el nodo que se ha movido de posición en la red.

OPCIÓN 2: También pueden usarse los propios paquetes que circulan por la red para confirmar la presencia de un vecino o detectar que un nodo ya no está en su posición inicial. En este sentido, una **alternativa mixta** al caso anterior sería la siguiente:

Cada nodo guarda en la caché tabla con nodos vecinos de los que recibe o a los que envía paquetes de forma exitosa (no es incompatible con guardar además rutas hacia los nodos origen o destino de los paquetes que circulan por él). Esta tabla tiene un tiempo de validez determinado y el envío del mensaje de control HELLO también tiene un intervalo de periodicidad. La idea es enviar HELLO únicamente a los nodos de los cuales no ha tenido conocimiento desde la última vez que lo envió, es decir, que solamente respondan aquellos que han estado callados en el intervalo desde el último envío del mensaje HELLO.

Para ello, dado que este mensaje lo escucharán todos los vecinos que tengan una adecuada calidad de enlace, HELLO debería incluir un campo con los ID de los destinatarios (o vacío si se quiere enviar a todos ellos). De esta forma ahorramos

respuestas de los demás nodos no incluidos como destinatarios. Las tres maneras de conocer a un vecino son:

1. El nodo vecino ha enviado un mensaje de HELLO
2. El nodo vecino ha enviado un mensaje de CHREQ, CHREP, CHACCEPT, CHDEL o CHROUTE (más adelante se detallarán estos mensajes)
3. El nodo vecino ha enviado paquetes de datos

OPCIÓN 3: No utilizar paquetes HELLO e ignorar todos aquellos paquetes de control cuyo destinatario no sea el nodo que lo recibe. De esta forma se ahorra no sólo energía, sino también memoria y capacidad de procesado ya que un nodo de la red no tiene que procesar los paquetes recibidos si no es él el destinatario (ya no se guardan datos en la caché ni se procesan los mensajes para comparar los datos contenidos con los de las tablas del nodo receptor).

En este caso, si la causa de desaparición del nodo es que se ha movido, al nodo vecino le llegará un mensaje CHDEL procedente del antiguo hijo a través del nuevo padre que tenga en la red. Si la causa es que se ha apagado o que el enlace está temporal o definitivamente inutilizable, no se detectará hasta que exista algún paquete de datos que deba pasar por el nodo desparecido. En este caso, el nodo que le envía el paquete (y que por tanto no recibe respuesta) deberá generar un mensaje de error en el envío del paquete (ROUTEERROR) y reenviarlo en la dirección por la que le llegó el mismo. Tras reintentarlo varias veces sin tener éxito, el paquete acabará descartándose pero en la tabla seguirá existiendo esa entrada.

Con respecto a esta última afirmación, el nodo vecino (padre o del mismo nivel) no tiene a priori ningún mecanismo para saber si ese nodo se ha apagado o simplemente el enlace está temporalmente fuera de servicio (aunque sí detectaría que se ha movido y, por tanto, borraría la ruta antigua ya que le llegará un mensaje CHDEL). En consecuencia, si el nodo se ha apagado definitivamente esa ruta seguiría en la memoria de los nodos para siempre. Para solucionarlo, deberá

llevarse un contador en el nodo vecino con el número de ROUTEERROR generados (uno cada vez que intente enviarle un paquete sin éxito). Al tercer ROUTEERROR se borra esa entrada de la tabla y se genera un CHDEL para que todos los nodos que contengan esa ruta también la borren de forma definitiva.

Comparando cada una de las alternativas expuestas, la que puede intuirse a priori que genera **un mayor rendimiento es la OPCIÓN 3**, y es por tanto esta opción la implementada en los Diagramas de Flujo.

1.b) Actualización de rutas por desaparición de nodo con hijos en la red

En este caso pueden darse varias situaciones, que se contemplan en el proceso *"Búsqueda de rutas alternativas"*:

1.b.1) No cambia la estructura del árbol jerárquico

Al desaparecer el nodo D la estructura y los niveles de los demás nodos siguen siendo los mismos.

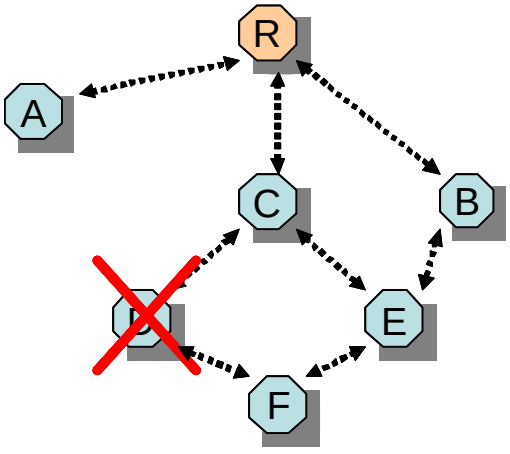


Figura 14. Desaparición de nodo con hijo sin cambio de estructura

1.b.2) Sí cambia la estructura del árbol jerárquico

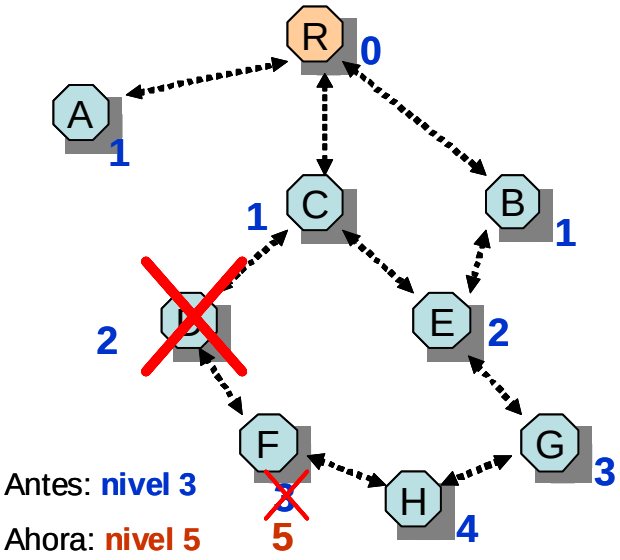


Figura 15. Desaparición de nodo con hijo con cambio de estructura

Al desaparecer el nodo D sí cambia la estructura del árbol ya que su hijo F se queda *huérfano* y tiene que buscar otros padres a los que asociarse.

En este caso particular, el nodo H que antes era hijo de F ahora se convierte en su padre.

También puede ocurrir que un nodo que antes era vecino del mismo nivel ahora se convierta en el padre. Para ello, el nodo F (y también el C) deben detectar que el nodo D ya no está (o que su enlace con él es de pésima calidad y por tanto no puede utilizarse). En función de la opción elegida en el apartado 1.a, sólo lo detectarán cuando tengan que enviarle un paquete de datos.

1.b.3) Algún nodo se queda huérfano

Al desaparecer el nodo D hay una parte de la red que se queda sin conexión a la misma (nodos F, G y H). Lo único que puede hacerse es que estos nodos sigan formando entre ellos una red diferente de la principal, cuyas tablas de rutas contengan a todos sus vecinos y nodos de esta red aislada, para de esta forma permitir el intercambio de paquetes de datos entre los nodos de que han quedado aislados de la red principal.

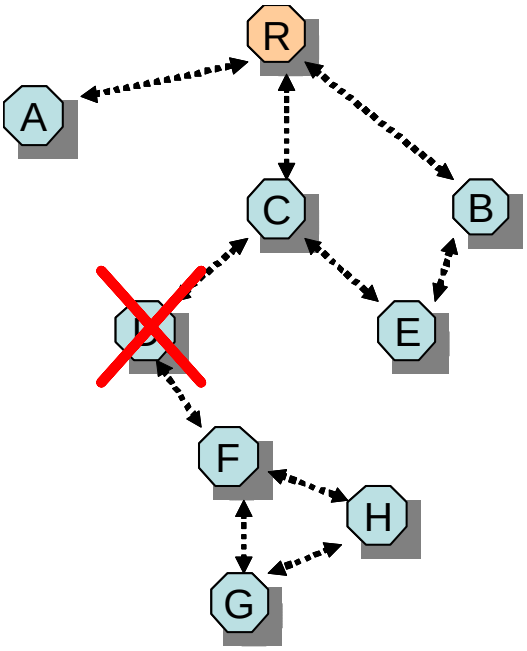


Figura 16. Nodos huérfanos

En este caso, el nodo F al detectar que su padre ya no está debería buscar otro nuevo padre (de la misma forma que en el caso 1.b.2) para poder seguir conectado a la red.

En todas estas situaciones, cuando un nodo se asocia a otro padre porque ha perdido conectividad con el suyo, una vez asociado al nuevo padre deberá generar un mensaje *CHDEL* para que todos los nodos que las tuviesen en su tabla borren las rutas anteriores a través del padre antiguo y las sustituyan por el nuevo (Proceso *Búsqueda de Padre*). Por tanto, todos los paquetes que la red quiera enviar al nodo se enviarán por el/los nuevo/s padres, incluyendo F: si, por ejemplo, C ha estado un rato desconectado, es posible que no haya recibido estos paquetes CHDEL y por tanto siga creyendo que F es hijo suyo, por lo que le enviará mensajes. A la tercera vez (3 mensajes ROUTEERROR propios) procederá al borrado definitivo de la ruta en su tabla.

Estudio de un caso particular del 1.b.2 que da lugar al *Proceso de Inversión de Rama*

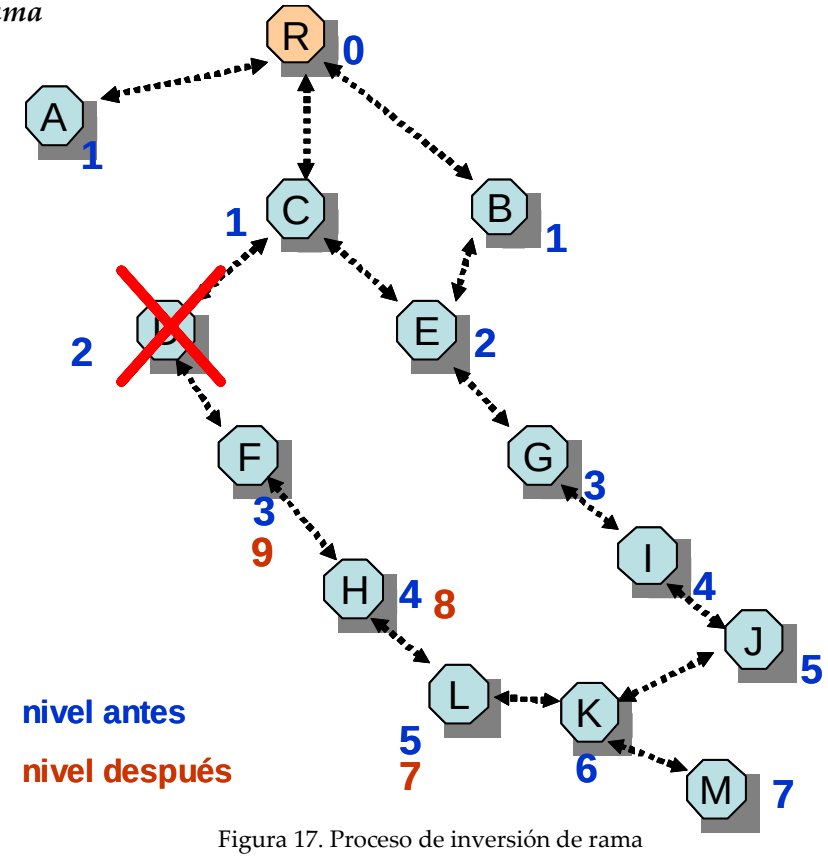


Figura 17. Proceso de inversión de rama

Como se ha indicado anteriormente, cuando F quiere mandar un mensaje a D y no puede conseguirlo, automáticamente se pone en marcha el proceso *"Búsqueda de rutas alternativas"*. En este caso, al no tener F más padres, se pone a su vez en marcha el proceso *"Búsqueda de padre"*.

Pero si al observar la figura adjunta, no basta con que F se asocie como hijo a H, que tiene nivel 4, ya que esa rama se quedaría sin poder enviar paquetes (no tienen nodos padre de niveles 3, 2 y 1). Para que exista coherencia de niveles, debería intentar buscar nuevas asociaciones por el otro lado de la rama de tal forma que se pueda reconstruir la rama de acuerdo a las normas del árbol jerárquico (debería quedar con los niveles que aparecen en marrón).

2) Pérdida de un enlace entre nodos

La forma de actuar y de tratar este caso es la misma que en el apartado 1.b, es decir, si un nodo no responde a un mensaje de HELLO o no recibe un paquete de datos, puede ser porque ese nodo no esté ahí, porque que se haya apagado o debido a que el enlace entre ambos sea de mala calidad o existan interferencias.

En el caso de que sea el enlace el que está inutilizable en ese momento, al no borrarse la entrada de la BD que contiene al nodo al cual no se puede llegar (sólo se borra si el nodo se mueve y se genera un mensaje CHDEL), transcurrido un tiempo cuando el enlace se recupere, no habrá que hacer nada y ese nodo ya podrá recibir mensajes.

Si mientras ese enlace está roto un hijo desea mandar sin éxito un mensaje, una vez que ha tratado de enviarlo varias veces procederá a asociarse como hijo a otro nodo. Cuando este enlace se restablezca y haya paquetes por enviar al antiguo hijo, éste los recibirá y detectará que han sido enviados desde el antiguo padre (con nivel menor que el que tiene ahora) y por tanto estudiará si asociarse de nuevo al antiguo padre (*Proceso de recepción de un paquete desde un antiguo padre, hijo o vecino*)

3) Actualización de rutas por aparición de nodo nuevo en la red

Tras el establecimiento del árbol jerárquico pueden existir nodos que no tengan asignado ningún padre y que consecuentemente deben asociarse a uno para poder enviar y recibir paquetes de datos. Esto puede ocurrir por tres motivos:

- El nodo sin padre **es un nodo nuevo o no estaba activo** en el momento de la construcción del árbol y sí lo está después.

- **Existían interferencias con otros nodos (colisiones de paquetes), tenía enlaces de mala calidad o estaba fuera de cobertura** en el intervalo tiempo en que se estableció el árbol jerárquico.

- El nodo **se estaba moviendo** mientras se establecía el árbol y, en consecuencia, no llegó a finalizar la asociación con ningún padre.

Otras veces un nodo se ve obligado a reconectarse a otro padre debido a la pérdida del enlace con el que se había asociado inicialmente.

Por último, también se contempla el **movimiento de un nodo**, que puede implicar la pérdida de conectividad con los nodos vecinos que tenía previamente, especialmente con los padres.

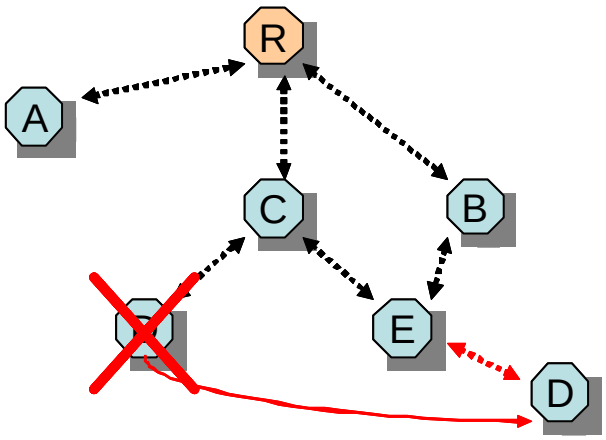


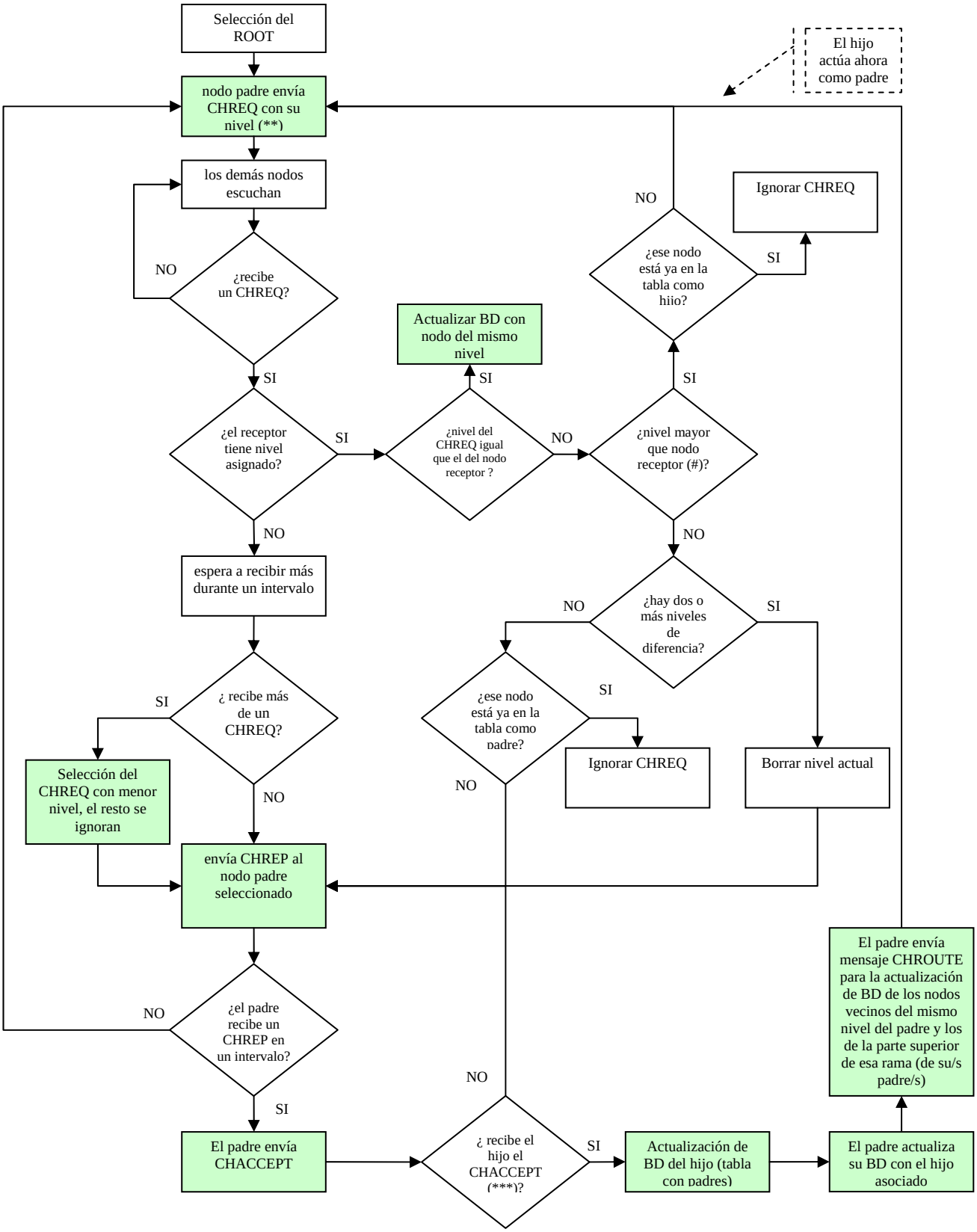
Figura 18. Movimiento de un nodo

En cualquiera de los casos anteriores el nodo debe asociarse a un padre, lo que se contempla en el *"Proceso Búsqueda de Padre"*

4.3.- Procesos que conforman el protocolo

A continuación se incluyen los Diagramas de Flujo de todo los procesos incluidos en el diseño del protocolo jerárquico con encaminamiento salto a salto.

Proceso de Construcción y Actualización Dinámica del árbol jerárquico



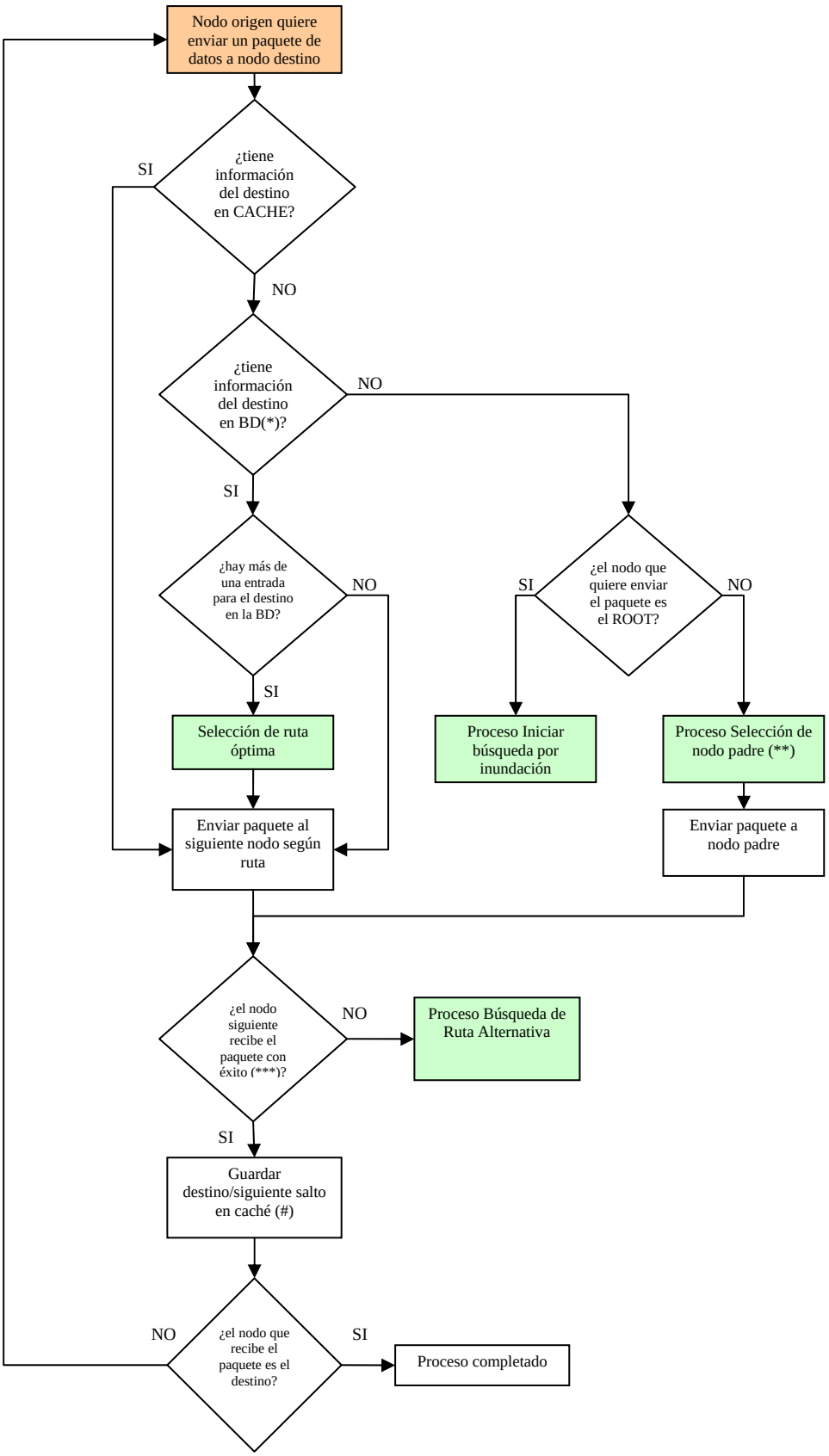
(*) Inicialmente los nodos no tienen nivel

(**) Si envía un CHREQ y no obtiene respuesta, repite el proceso una o dos veces y luego ya no lo sigue intentando

(***) De la misma forma, tras varios intentos sin que el padre envíe el CHACCEPT deja de enviar CHREP

(#) Si un nodo recibe un CHREQ con nivel mayor, lo que debe hacer es asociar a ese nodo como hijo y por tanto le envía CHREQ. Esto favorece la coherencia de niveles y evita que entre dos nodos vecinos exista una diferencia de dos o más niveles.

Proceso de Envío de un paquete de datos desde un nodo cualquiera de la red



(*) El que el nodo que quiere enviar el paquete tenga en su BD información del destino se debe a dos causas:

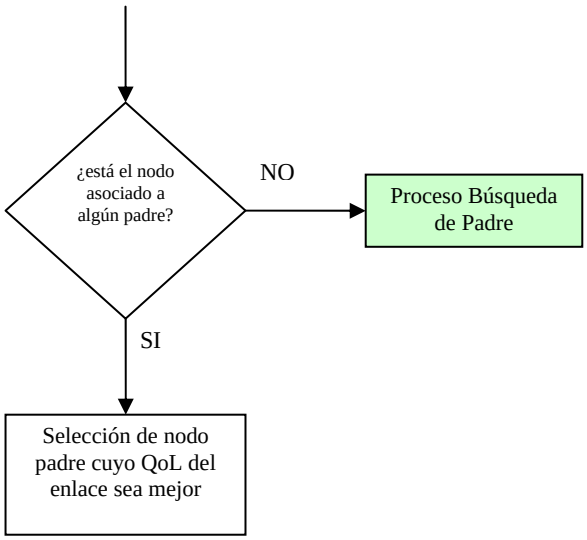
- El destino es un nodo vecino (del mismo nivel o de un nivel inmediatamente superior): en este caso sólo hay un salto entre origen y destino
- El destino se encuentra en alguna de sus ramas inferiores y, por tanto, envío el paquete al hijo/s que indique la BD

(**) Se supone que un nodo puede tener más de un padre. El parámetro utilizado para seleccionar el nodo padre a quien enviar el paquete es la **Calidad del Enlace (QoL)** entre ambos. Ver Proceso "Selección de Nodo Padre". Para enviar un paquete de datos es necesario haberse asociado como mínimo con un padre

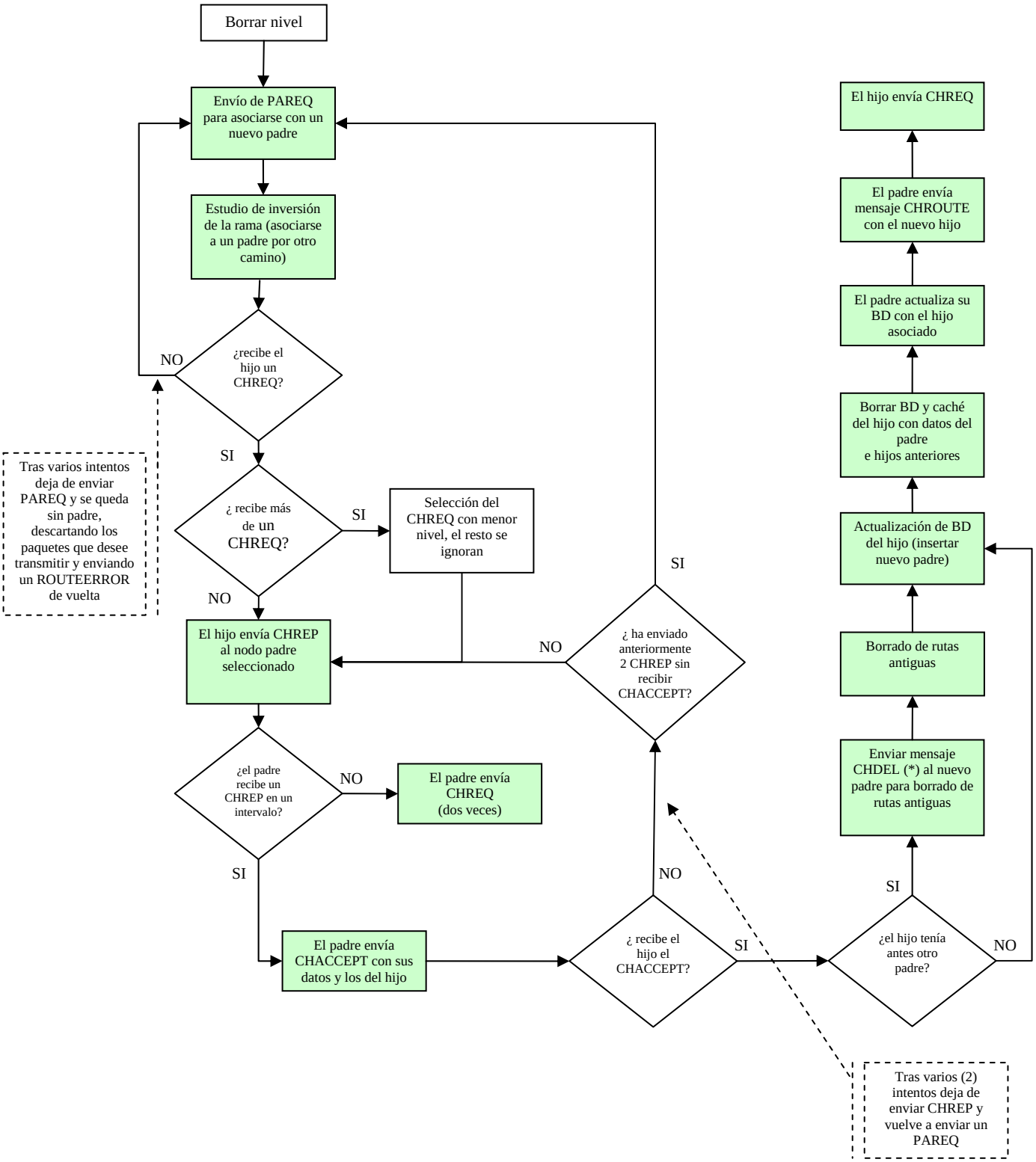
(***) ACK y retransmisiones: se ocupa el nivel de enlace

(#) De esta forma si se recibe a continuación otro paquete con el mismo destino (algo probable una vez que se ha recibido uno) se ahorra capacidad de procesamiento y energía ya que se sigue la ruta almacenada en la caché

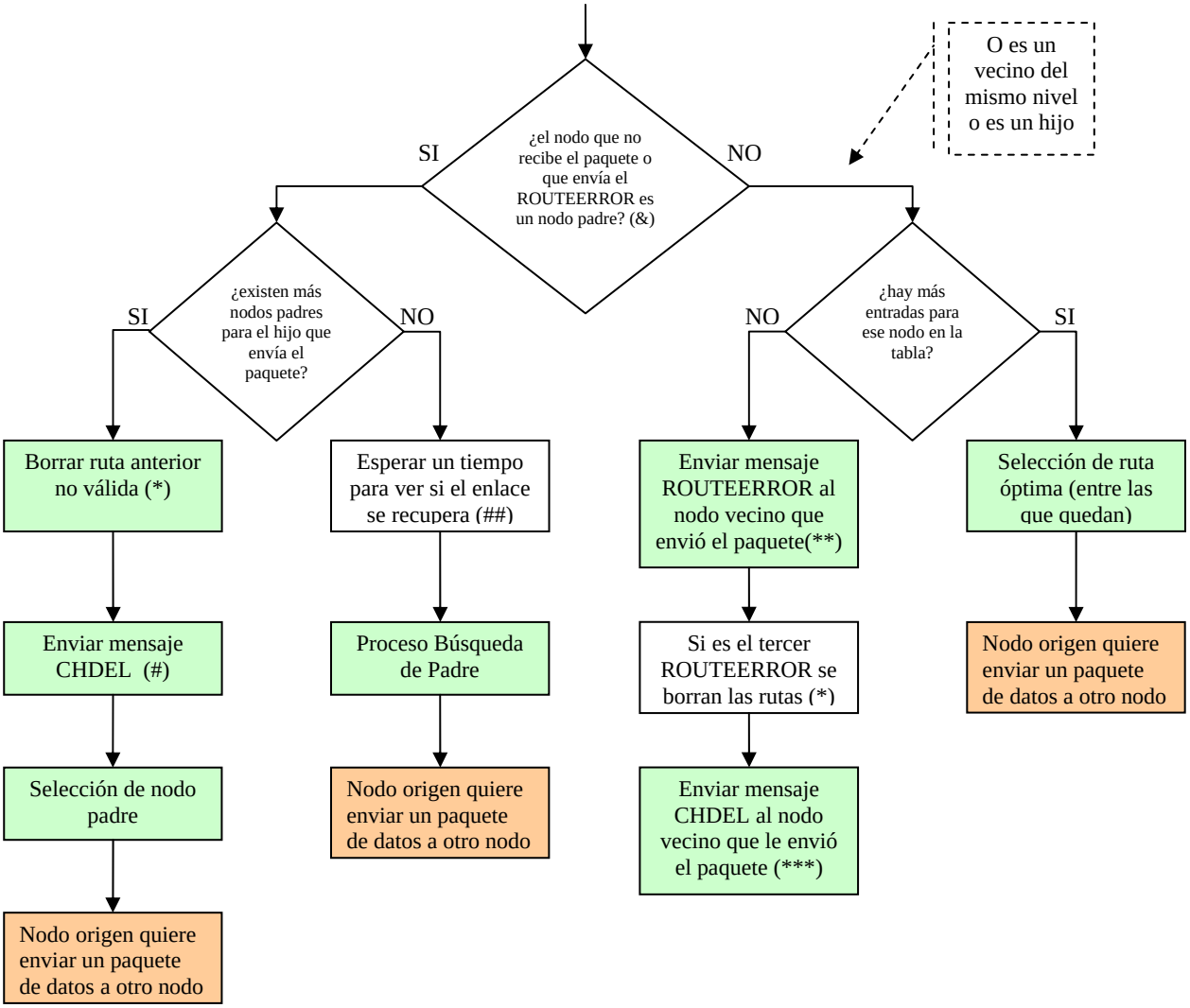
Proceso Selección de Nodo Padre



Proceso Búsqueda de Padre



Proceso Búsqueda de Ruta Alternativa



(&) Este proceso lo inicia el nodo origen cuando un nodo vecino no recibe el paquete que le ha enviado o cuando un nodo vecino le envía un ROUTEERROR

(*) Se borran todas las rutas en las que esté presente (la suya y la de todos sus hijos). Ver apartado 1.a de *"Actualizaciones de Rutas"*

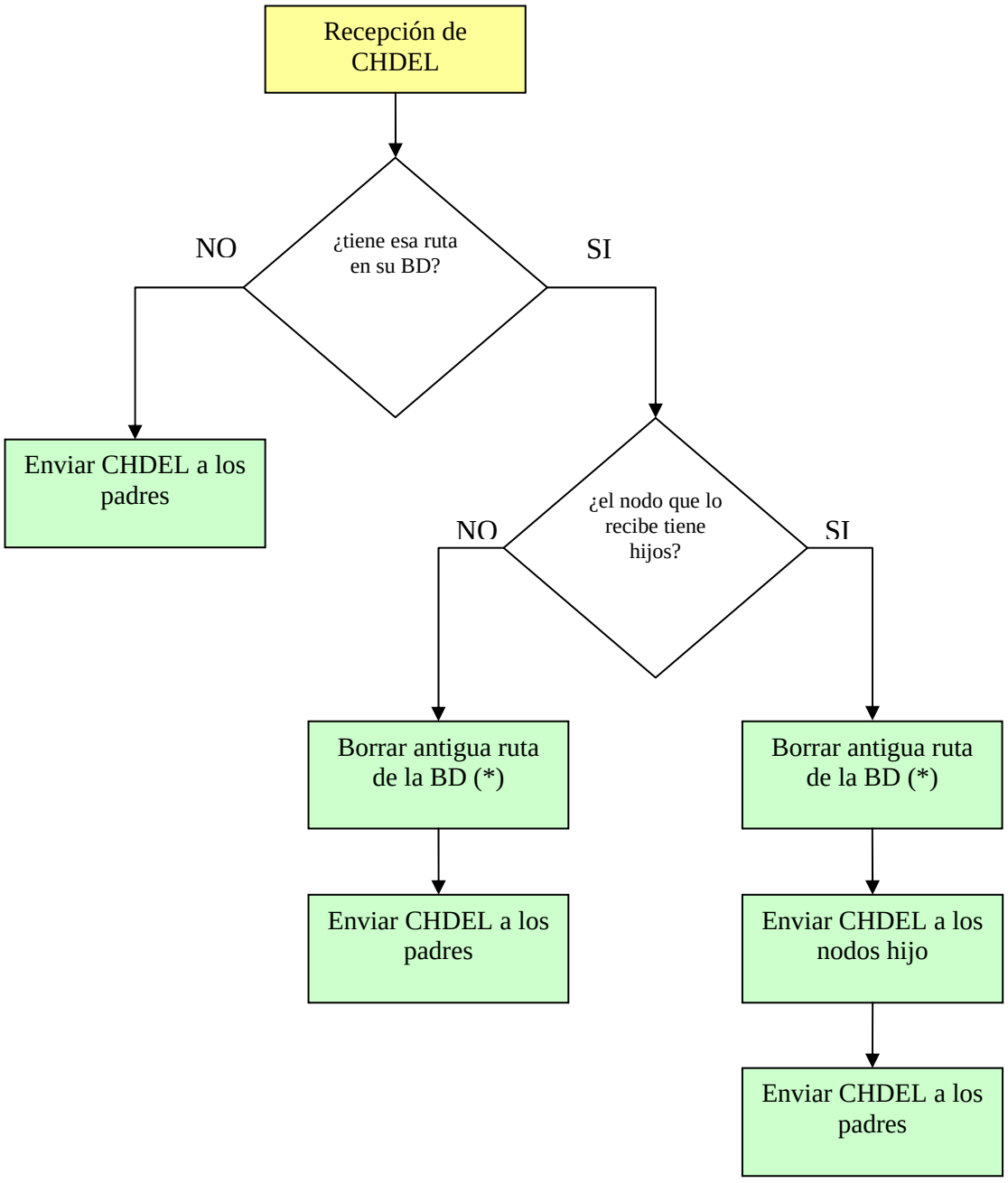
(**) Y si es el propio nodo el que ha generado el paquete a enviar no envía ningún ROUTEERROR sino que directamente borra la ruta y luego envía CHDEL a sus padres con la ruta a borrar

(***) Mensaje CHDEL para borrar la ruta antigua en aquellos nodos que la tuviesen. Ver proceso de *"Borrado de Rutas Antiguas"*

(#) Se puede guardar un contador con el número de veces que se intenta enviar un paquete a ese nodo y no contesta (mientras está el dato del otro nodo padre en la caché se seguirán enviando los paquetes por él, pero cuando desaparezca de la caché se volverán a intentar enviar por éste). En caso de que se siga sin poder (el enlace todavía no se ha recuperado o el nodo se ha apagado) se incrementa el contador y se busca otro nodo padre por el que enviarlo. Así hasta que número de contador = 3 (para que sea igual que los ROUTEERROR). Si en ese tiempo el enlace todavía no se ha recuperado, lo mas probable es que ya no lo haga y que el nodo se haya apagado o ya no esté allí, por lo que el hijo procederá a borrar ese nodo como padre y a enviar por otro nodo padre un mensaje CHDEL para que los demás también lo borren.

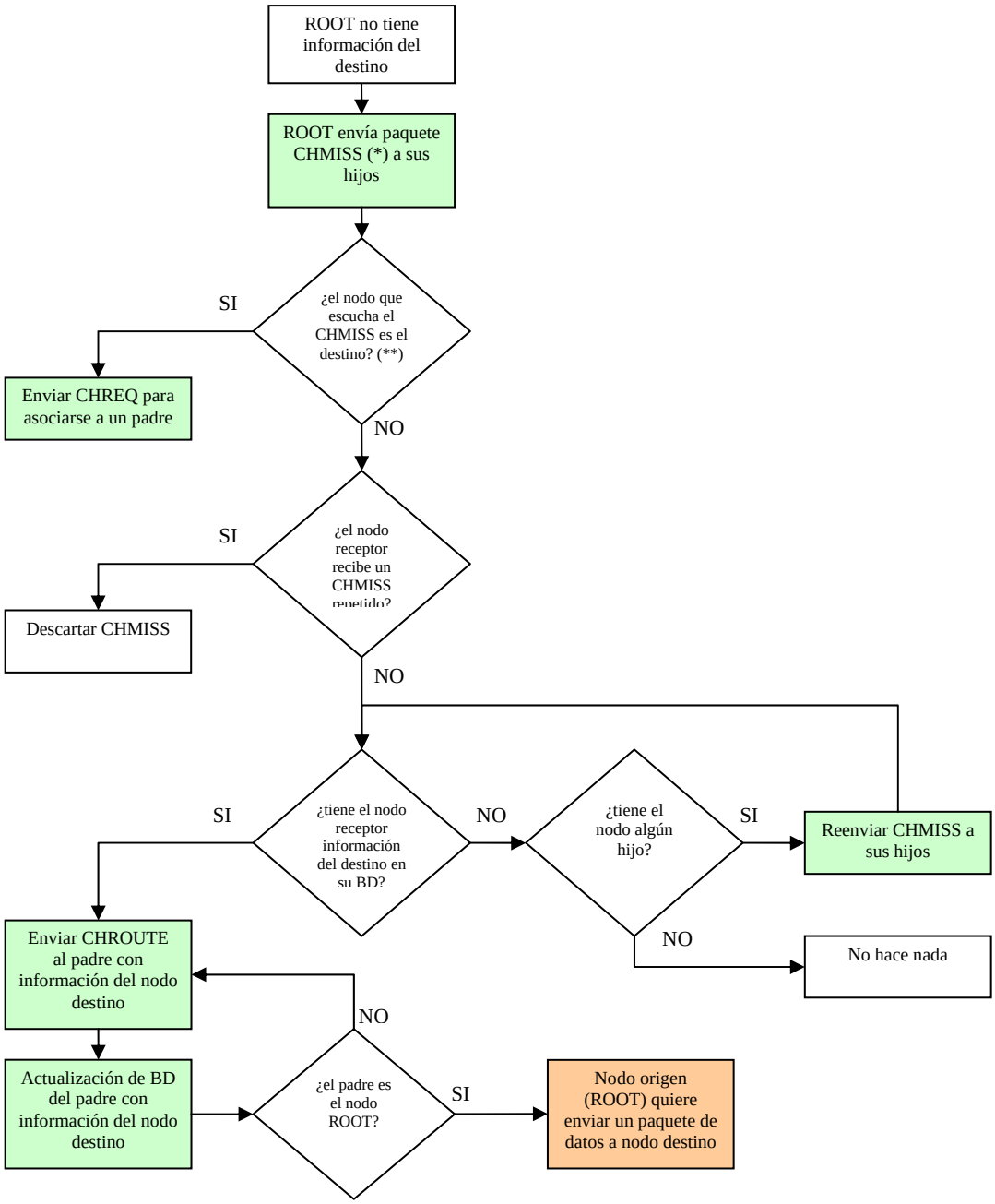
(##) Para lograr la coherencia con (#), primero se debe esperar a ver si el enlace se recupera y se pueden enviar los paquetes antes de iniciar el proceso de búsqueda de otro nodo padre. Si transcurrido un tiempo prudencial sigue sin recuperarse, entonces se da por perdido (se borra de la tabla de rutas) y se buscan otros padres.

Proceso Borrado de rutas antiguas



(*) Hay que borrar todas las rutas que contengan al nodo en cuestión

Proceso Iniciar búsqueda por inundación

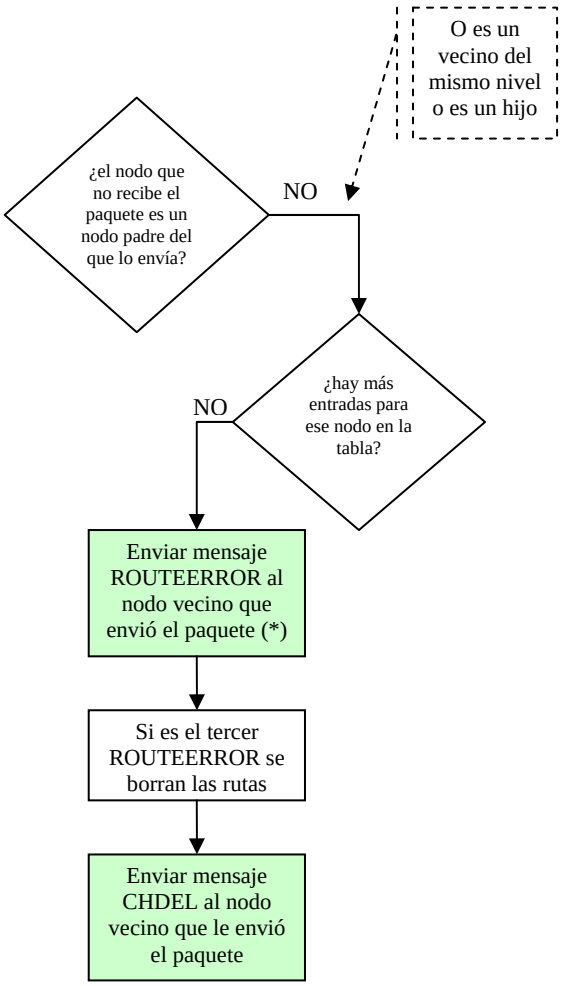


(*) CHMISS es un paquete de solicitud de información de la posición de un nodo en particular. Cuando un nodo lo reenvía, cambia en el paquete los valores de Nodo_origen y Nivel_nodo_origen y pone los suyos propios.

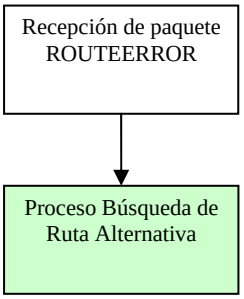
(**) Especialmente pensado para aquellos nodos en movimiento. Si no estuviese este recuadro, si un nodo está en movimiento puede que haya perdido la conectividad con sus nodos vecinos iniciales, por lo que no podría recibir paquetes de datos ni enviarlos hasta que no realice una nueva asociación con algún nodo padre. Tampoco serviría para nada la inundación de la red a través del paquete de búsqueda CHMISS para localizarle. Con este recuadro se permite que si el nodo destino escucha un CHMISS y él es el nodo buscado, entonces automáticamente busca una asociación con un padre para poder recibir el paquete de datos.
Cuando un nodo escucha un CHMISS, mira la dirección del nodo destino y la compara con la suya propia. Si no es él el destino, hace un *broadcast* del paquete CHMISS recibido. De esta forma se permite que un nodo que no tenga padre asociado en ese momento tenga que asociarse a uno para poder recibir el paquete

Proceso ROUTEERRROR

Generación del mensaje ROUTEERRROR (Contenido en *Proceso Búsqueda de Ruta Alternativa*)



Recepción del mensaje ROUTEERRROR



(*) Este mensaje UNICAMENTE lo enviará aquel nodo padre que tenga que enviar un paquete a uno de sus hijos del que sólo tenga una ruta (camino directo) y no pueda o que ya haya agotado las posibilidades de envío mediante otras rutas

Proceso Búsqueda de Ruta Alternativa:

(#) Se puede guardar un contador con el número de veces que se intenta enviar un paquete a ese nodo y no contesta (mientras está el dato del otro nodo padre en la caché se seguirán enviando los paquetes por él, pero cuando desaparezca de la caché se volverán a intentar enviar por éste). En caso de que se siga sin poder (el enlace todavía no se ha recuperado o el nodo se ha apagado) se incrementa el contador y se busca otro nodo padre por el que enviarlo. Así hasta que número de contador = 3 (para que sea igual que los ROUTEERROR). Si en ese tiempo el enlace todavía no se ha recuperado, lo mas probable es que ya no lo haga y que el nodo se haya apagado o ya no esté allí, por lo que el hijo procederá a borrar ese nodo como padre y a enviar por otro nodo padre un mensaje CHDEL para que los demás también lo borren.

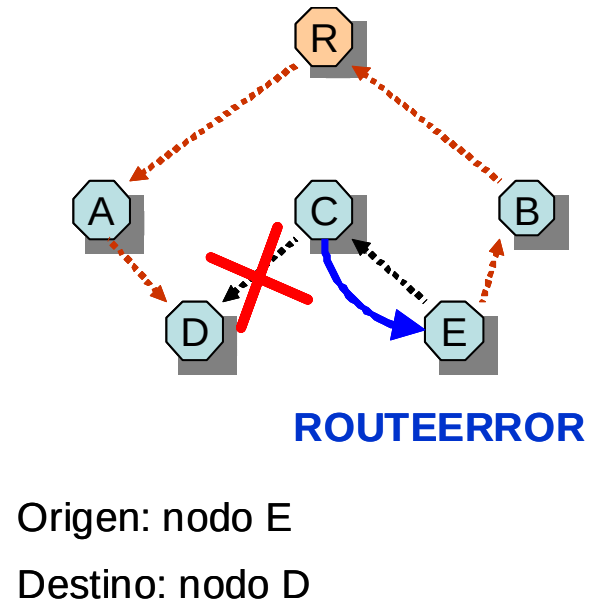
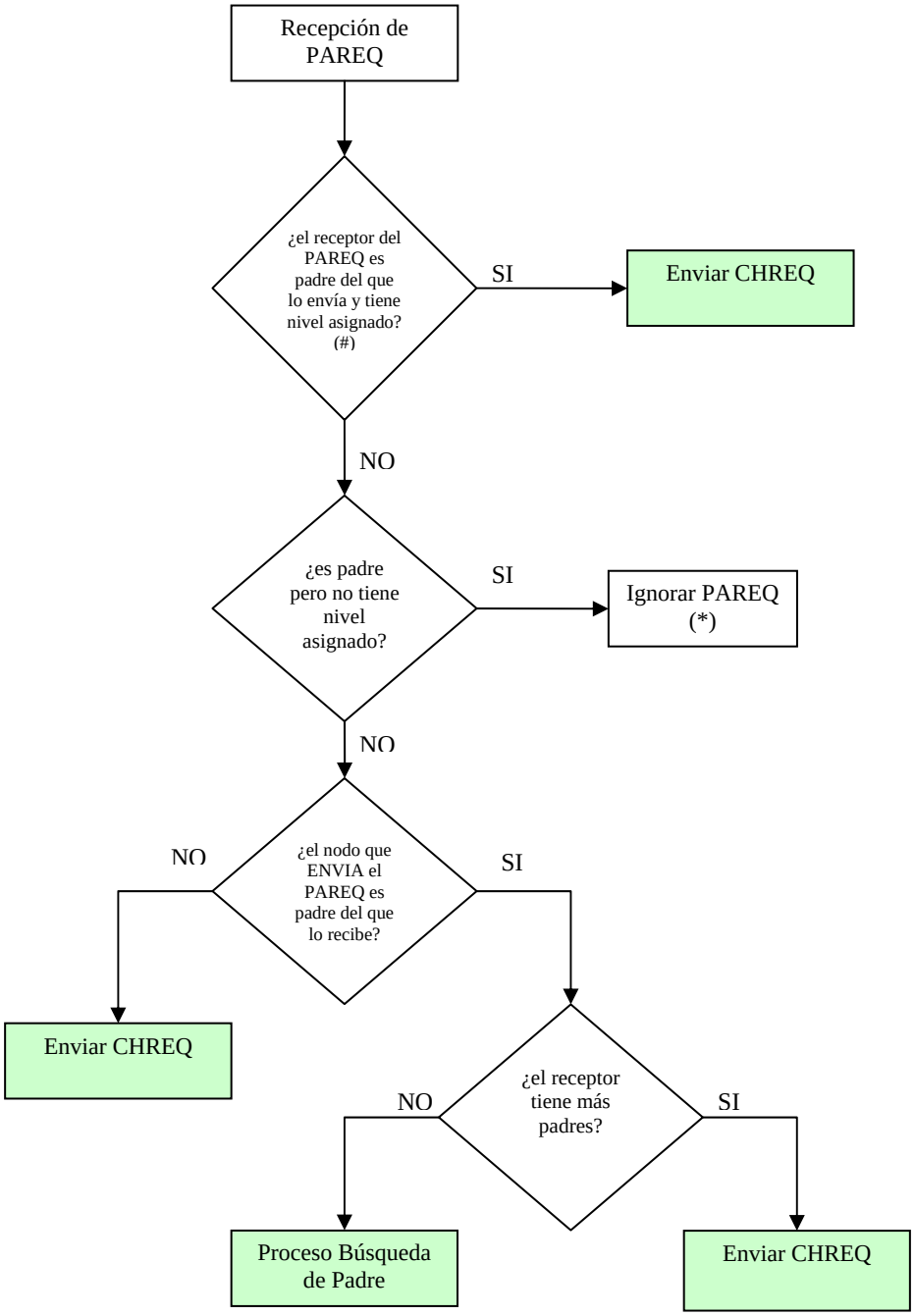


Figura 19. Error de ruta

Proceso Estudio de Inversión de Rama



Este proceso favorece que la estructura del árbol jerárquico cambie dinámicamente y se adapte a las circunstancias de la red: pérdidas de enlaces, movimientos de nodos, apagados, etc.

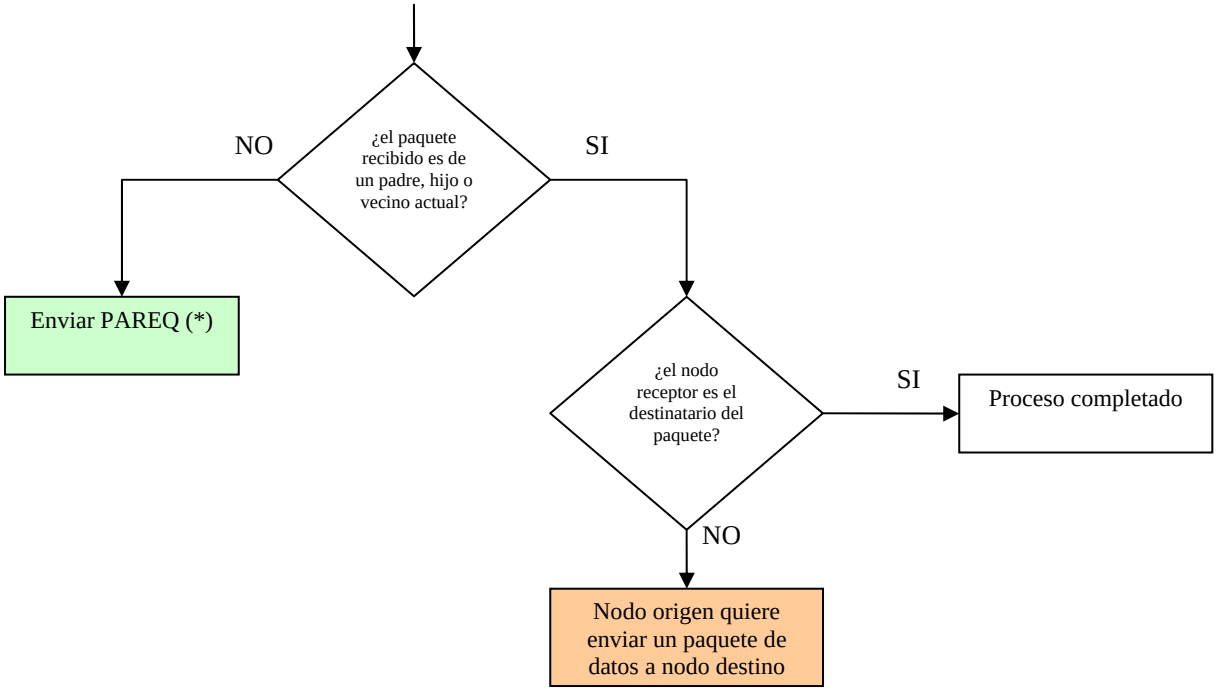
Cuando un nodo pierde las conexiones con sus padres y no es capaz de enviar un paquete de datos, automáticamente se buscan nuevos padres por donde poder enviar los mensajes. Sin embargo, en ocasiones es necesario invertir el comportamiento de una rama, es decir, que los padres sean ahora los hijos y los hijos se comporten como padres.

(*) Si hay que invertir la rama, el hijo anterior se convertirá ahora en padre, pero para ello deberá enviar a su vez un PAREQ (*"**PA**rent **REQ**uest"*) (una vez que reciba el del antiguo padre). En ese caso el antiguo padre debe ignorarlo.

Con respecto al padre del nodo que envía el PAREQ, si el nodo envía este mensaje es porque no ha podido enviarle un paquete de datos (el padre no le escucha, con lo que tampoco escuchará el PAREQ).

(#) Se corresponde con el envío de un antiguo padre que se ha desconectado temporalmente y no se ha enterado (no ha recibido ningún CHDEL) de que su hijo se ha asociado con otro u otros padres

Recepción de un paquete de datos de un antiguo padre, hijo o vecino



(*) Como el nodo no sabe si es un antiguo padre o un antiguo hijo ya que ha borrado esa entrada de su tabla, lo que hace cuando reciba un paquete de un nodo que no tiene en su tabla como vecino, padre ni hijo es enviarle un PAREQ. La explicación es la siguiente:

> En los procesos anteriores se ha mostrado cómo debe actuar un nodo cuando desea enviar un paquete y el padre con el que está asociado (o uno de ellos) no lo recibe correctamente. Esto se puede deber a varias causas, como ya se ha indicado:
>
> - El nodo padre se ha apagado de forma definitiva o se ha movido
> - El nodo padre se ha apagado temporalmente o el enlace entre ambos estaba temporalmente fuera de servicio
>
> En ambos casos, el nodo (tras un tiempo de espera si tiene únicamente un padre o tras enviarle varios paquetes sin éxito si tiene más de uno) el nodo opta por asociarse a nuevos padres que le permitan el envío y la recepción de paquetes.
>
> Pues bien, si se estudia en detalle el segundo caso puede ocurrir que, tras ese proceso y después de que el nodo haya cambiado de padres, **el nodo que antes era padre** y que era temporalmente inalcanzable, ahora sí esté en disposición de recibir y enviar paquetes (por ejemplo, porque ese enlace se ha recuperado). A este nodo no le llegará ningún paquete con destino a su antiguo hijo porque éste, una vez se ha asociado a otro padre, genera un mensaje CHDEL para que los demás nodos borren la ruta que le asociaba con su antiguo padre. En consecuencia, existe sólo una posibilidad de que el antiguo padre le envíe un paquete al antiguo hijo, que es que lo genere él mismo y el destino sea precisamente su hijo; en ese caso lo hará porque todavía cree que lo es.

Para que exista coherencia con los procesos definidos anteriormente y que el árbol jerárquico se reconstruya dinámicamente, **en este caso el antiguo hijo deberá asociarse de nuevo al padre** (siempre que el nivel del mismo sea menor o igual que los niveles de los nuevos padres a los que se ha asociado), para lo cual enviará un PAREQ que en ese caso particular tendrá como respuesta un CHREQ.

Además, el hijo deberá refrescar la base de datos del padre con sus entradas, ya que es probable que la estructura de la rama inferior haya cambiado si hay movimientos y el nodo padre que estaba temporalmente sin conexión a la red no se haya enterado (no ha recibido los CHDEL asociados con los cambios de padre).

En el caso de que sea un **antiguo hijo** el que envía paquete (es decir, que el padre ha intentado enviarle varias veces datos sin éxito y finalmente lo borró como hijo), deberá asociarlo de nuevo, para lo cual le envía primero un PAREQ y a continuación le enviará un CHREQ.

En el caso de un **vecino** ocurre lo mismo ya que si se le ha enviado varias veces un paquete sin éxito habrá sido borrado de la tabla. En este caso bastaría con actualizar la tabla añadiendo una entrada para el nodo vecino.

Proceso de Selección de Ruta Óptima

Mediante el campo QoL *(Quality of Link)* un nodo almacena la calidad del enlace con un nodo vecino en particular. Existen varias opciones:

1. Almacenar solamente el estado del enlace con el nodo vecino (hijo, padre o vecino directo), sin tener en cuenta los enlaces entre los nodos intermedios, en caso de almacenar una ruta de un nodo que tiene como mínimo dos niveles más

2. Tener en cuenta todo el enlace y almacenar los estados de todos los enlaces hasta llegar al destino. El inconveniente de esta opción es que sería necesario **actualizar estos datos** cuando el estado de un enlace cambie de una manera más o menos significativa, con el consiguiente envío de paquetes de información actualizando el valor del nuevo estado hasta llegar al *Root*. En este caso también se podía incluir otro campo en la tabla indicando el nº de saltos hasta el destino. En el caso particular mostrado, la tabla del *Root* sería:

NODO	ADDR	QoL	LEVEL	INTER	LV INT	PADRE
R	A	0.5	1	-	-	NO
	B	0.4	1	-	-	NO
	F	0.3	1	-	-	NO
	C	0.5+0.4	2	A	1	NO
	C	0.4+0.7	2	B	1	NO
	D	0.5+0.4+0.8	3	A	1	NO
	D	0.4+0.7+0.8	3	B	1	NO
	E	0.5+0.4+0.9	3	A	1	NO
	E	0.4+0.7+0.9	3	B	1	NO

Tabla 2. Caso particular de tabla de encaminamiento del *Root*

En el caso de que puedan existir bucles, a este nodo (*Root* en este ejemplo) le pueden llegar dos o más mensajes de rutas en el que el único campo diferente sea precisamente QoL. *Root* entonces no debería almacenar sin más dicha ruta (no aporta información de encaminamiento adicional), sino que debería utilizarla para el cálculo de la ruta óptima. Para ello, debe comparar el valor indicado en este campo con la que tiene en su tabla. Si este valor es mejor (con lo que la ruta que le llega tiene mayor calidad de enlace), sustituye la que tiene en su tabla por la nueva. Si este valor es peor (la ruta que le llega es de peor calidad que la que tiene almacenada) entonces descarta la nueva ruta y no modifica su BD.

El cálculo de la ruta óptima se haría escogiendo, de entre todas las rutas que tengan al nodo destino en el campo ADDR (ADDRESS), aquella que tenga mejor valor de QoL.

4.4.- Estructura de los Mensajes de Control y de la Tabla de Rutas

4.4.1.- Mensajes de Control

En los diagramas anteriores se han incluido una serie de paquetes de control cuya finalidad es facilitar las acciones a realizar por parte de los nodos para que estén en disposición de poder enviar un paquete de datos cuando así lo requieran.

A continuación se describe la función y los campos que debe tener cada uno de ellos:

- **CHREQ, PAREQ:** ambos son paquetes de *broadcast*. La función de CHREQ es preguntar si en su entorno hay algún nodo que quiera

asociarse a él como hijo. Por el contrario, PAREQ consulta si hay algún nodo que pueda acogerle a él como hijo

Campo *Valor del campo*	**Nodo_origen**	**Nivel_nodo_origen**
	id_node_origin	level_origin

Tabla 3. Estructura CHREQ, PAREQ

- **CHREP:** su misión es la de responder tras recibir un CHREQ para asociarse como hijo al nodo que lo envió

Nodo_destino (padre)	**Nivel_nodo_destino (padre)**	**Nodo_origen (hijo)**	**Nivel_nodo_origen (hijo)**
id_node_parent	level_parent	id_node_son	level_son

Tabla 4. Estructura CHREP

- **CHACCEPT:** este paquete lo genera el nodo padre y lo envía al nodo hijo tras recibir un CHREP. Su función es la de confirmar al nodo como hijo

Nodo_destino (hijo)	**Nivel_nodo_destino (hijo)**	**Nodo_origen (padre)**	**Nivel_nodo_origen (padre)**
id_node_son	level_son	id_node_parent	level_parent

Tabla 5. Estructura CHACCEPT

- **CHROUTE:** su misión es la de informar a lo nodos superiores del árbol del establecimiento de rutas hacia nuevos nodos. La genera un nodo cuando asocia a un hijo y sus campos son algunos de los incluidos en la tabla de encaminamiento (RT, *routing table*), excepto la información del nodo intermedio (que no se usa) y a los que se añade al principio los datos del nodo que envía el paquete y que se convierte por tanto en nodo intermedio

Nodo_origen	Nivel_nodo_ origen	Nodo a incluir	Nivel del Nodo a incluir	Calidad del enlace hasta el nodo
Id_node_origin	level_origin	RT_node	RT_level	RT_QoL

Tabla 6. Estructura CHROUTE

- **CHDEL:** este paquete indica que todas las rutas que contengan al nodo que se incluye en el paquete deberán ser borradas en la tabla del nodo que lo reciba

Nodo_destino	Nivel_nodo_d estino	Nodo a borrar	Nivel del Nodo a borrar	Calida del enlace hasta el nodo
Id_node_origin	level_origin	RT_node	RT_level	RT_QoL

Tabla 7. Estructura CHDEL

- **CHMISS:** este paquete lo genera el nodo *Root* y se propaga por la red cuando no se encuentra una ruta válida para un nodo en particular. Cada nodo que lo recibe incluye sus datos en el paquete y lo reenvía hasta que se encuentre información válida sobre el nodo buscado

Nodo_destino	Nodo_origen (padre)	Nivel_nodo_origen (padre)
Id_node_origin	id_node_parent	level_parent

Tabla 8. Estructura CHMISS

- **ROUTEERROR:** este paquete de control se genera cuando no se puede enviar un paquete de datos según la ruta indicada porque existe algún problema

Nodo_destino	Nivel_nodo_destino	Nodo_origen	Nivel_nodo_origen	Nodo_no alcanzable
id_node_parent o id_node_son o id_node (vecino)	level_parent o level_son o level (vecino)	Id_node_origin	level_origin	Id_node

Tabla 9. Estructura ROUTEERROR

4.4.2.- Estructura de la tabla de rutas

Como hemos mencionado a lo largo de este trabajo, todos los nodos de la red contendrán una RT en la que almacenarán todos los datos necesarios para el envío de paquetes a lo largo de la red según indica el protocolo descrito.

La estructura y campos propuesta para esta tabla de rutas es la que se indica a continuación:

NODO	ADDR	QoL	LEVEL	INTER	LV INT	PADRE
R	A	0.5	1	-	-	NO
	B	0.4	1	-	-	NO
	F	0.3	1			NO
	C	0.5+0.4	2	A	1	NO
	C	0.4+0.7	2	B	1	NO
	D	0.5+0.4+0.8	3	A	1	NO
	D	0.4+0.7+0.8	3	B	1	NO
	F	0.5+0.4+0.9	3	A	1	NO
	F	0.4+0.7+0.9	3	R	1	NO

Tabla 10. Estructura propuesta para tabla de rutas

La descripción de los campos es la siguiente:

- *RT_node*: id del nodo del que se almacena su ruta en la tabla — ADDR
- *RT_level:* nivel del nodo del que se almacena su ruta en la tabla — LEVEL
- *RT_QoL:* calidad acumulada de la ruta hasta el nodo almacenado — QoL
- *RT_node_int:* siguiente nodo de la ruta hacia el nodo almacenado — INTER
- *RT_level_int:* nivel del siguiente nodo de la ruta hacia el nodo almacenado — LV INT
- *RT_flag_parent:* indica si el nodo almacenado es padre (YES/NO) — PADRE

Para finalizar este apartado se indicarán las operaciones que se pueden efectual sobre la tabla de encaminamiento:

❖ **Añadir una ruta hacia un nodo:**

ADD(RT_node, RT_level, RT_QoL, RT_node_int, RT_level_int, RT_flag_parent)

❖ **Borrar una ruta que contenga a un nodo:**

DEL(RT_node, RT_level, RT_QoL, RT_node_int, RT_level_int, RT_flag_parent)

❖ **Buscar si un nodo está contenido en la tabla:**

FIND(nodo, nivel)

Respuesta de la función:

find = YES / NOT

4.5.- Programación del algoritmo de Construcción del Árbol Jerárquico

El objetivo principal de esta parte final del PFM ya se ha explicado en el *Apartado 3. Metodología* del proyecto. En esta última etapa se procederá a realizar una programación básica de uno de los módulos principales del algoritmo de forma que se pueda entender más en profundidad se funcionamiento y sirva de interfaz para una futura programación del mismo en un lenguaje más específico.

Para esta última etapa del PFM se ha seleccionado para su programación por su relevancia el módulo correspondiente a la "*Construcción y actualización dinámica del árbol jerárquico*".

Para realizar esta programación se ha dividido el algoritmo que lo implementa en parte, de la misma forma que se dividió este módulo en procesos más específicos. Por tanto, se tendrá por una parte lo que podría denominarse el *programa* o la *función principal*, denotada por BUILDING_HT().

En torno a esta parte principal del código se realizan llamadas a otras funciones. Estas otras funciones también se han dividido en funciones principales y en secundarias. Las principales serán las que implementen los paquetes de control más importantes (CHREQ, CHREP, CHACCEPT) y las secundarias, el resto.

A continuación se incluye cada una de estas tres partes del código:

CONSTRUCCIÓN Y ACTUALIZACIÓN DINÁMICA DEL ÁRBOL JERÁRQUICO: PROGRAMA PRINCIPAL

/* definición de variables

id_node: identificador del nodo

id_node_origin: identificador del nodo que envía el paquete y que se incluye en los mensajes a enviar

id_node_parent: identificador del nodo padre

level: número de nivel del nodo

level_parent: número de nivel del nodo padre

level_origin: es el nivel del nodo que envía el paquete y que se incluye en los mensajes a enviar

level_origin_min: en caso de recibir varios CHREQ, se selecciona el de menor nivel, valor que se asocia a esta variable

num_CHREQ_sent: número de veces que un nodo envía un paquete CHREQ (si lo envía 3 veces y no obtiene respuesta, deja de enviarlo)

num_CHREQ_rec: número paquetes CHREQ que un nodo recibe en un intervalo (se selecciona el de menor nivel)

num_CHREP_sent: número de veces que un nodo envía un paquete CHREP (si lo envía 3 veces y no obtiene respuesta, deja de enviarlo)

num_CHACCEPT_sent: número de veces que un nodo envía un paquete CHACCEPT (si lo envía 3 veces y no obtiene respuesta, deja de enviarlo)

flag_parent: indica si el nodo tiene padre. Sus posibles valores son:

0 : no tiene padre asociado

1 : sí tiene padre asociado

2 : el nodo es el ROOT

find: respuesta que devuelve la llamada a la función FIND

*/

```
//    CONSTRUCCIÓN DEL ÁRBOL JERÁRQUICO

void BUILDING_HT()
{

//    inicialización de variables

num_CHREQ_sent = 0
num_CHREQ_rec = 0
num_CHREP_sent = 0
num_CHACCEPT_sent = 0

if NODO = ROOT then
        flag_parent = 2
        level = 0
else
        flag_parent = 0
        level = {vacío}
endif

//    GENERACIÓN DE MENSAJES, EMPEZANDO POR ROOT

if flag_parent = 2 then
        id_node_origin = id_nodo
        level_origin = level      // el nivel que introduzco en el CHREQ es el nivel que tiene el ROOT (nivel=0)
        call BROADCAST_CHREQ(id_node_origin,level_origin)      // llamada a la función que genera el mensaje CHREQ con los parámetros id y nivel del root y espera un CHREP
endif

//    RECEPCIÓN DE LOS PAQUETES CHREQ, CHREP Y CHACCEPT, INCLUSIÓN EN TABLAS Y ENVÍO DE CHROUTE

while CHREQ recibido do
```

```
if flag_parent = 1 then          // cuando se recibe un CHREQ y ya se tiene padre asociado, hay que estudiar si es un posible hijo, un vecino o un nuevo padre y asociarse él o almacenarlo en la tabla de rutas
        if level = level_origin then      // si es un vecino comprueba si lo tengo en la tabla y, si no es así, lo introduce
                call FIND(id_node_origin, level_origin)
                if find = NOT then
                        call ADD(id_node_origin, level_origin, QoL, , , NO)
// añade el vecino a la tabla
                        call SEND_CHROUTE(id_node, level, id_node_origin, level_origin, QoL)
// envía la información de ruta hacia su nuevo vecino hacia arriba
                endif
        else
                if level < level_origin then      // estudia si puede ser un nuevo hijo
                        call FIND(id_node_origin, level_origin)
                        if find = NOT then
                                call BROADCAST_CHREQ(id_node, level)      // el que recibe el CHREQ es ahora el que lo envía
                        endif
                else
                        if level >=      level_origin - 2 then    // si hay más de dos niveles de diferencia debe asociarse al nodo origen como nuevo padre y desasociarse de los demás padres que tenía
                                call SEND_CHDEL(id_node, )          // envía mensaje CHDEL para borrado de rutas a través de sus antiguos padres hasta él (borrado de todos sus padres anteriores)
                                level = level_origin + 1          // actualiza su nuevo nivel
                                call DEL( , , , , , YES)          // borra de su tabla a sus padres actuales
                                call SEND_CHREP(id_node_origin, level_origin, id_node, level)
// no es broadcast, tiene un destinatario (el resto de nodos ignoran el CHREP). Además de enviar un CHREP espera un CHACCEPT
                                else
                                        call FIND(id_node_origin, level_origin, YES)    // busca en la tabla si ese nodo ya existe como padre
                                        if find = NOT then
                                                level = level_origin + 1
```

```
                                        call SEND_CHREP(id_node_origin, level_origin, id_node, level)
// si no existe como padre le responde con un CHREP
                                    endif
                                endif
                            endif
                        endif
        else    // cuando se recibe un CHREQ y no se tiene padre asociado
                call GUARDAR_CHREQ(id_node_origin, level_origin)        // espero a recibir varios CHREQ
en un intervalo
                if num_CHREQ_rec > 1 then
                        call SELECT_CHREQ()
                        for (id_node_origin,level_origin_min) in SELECT_CHREQ() do      //  la  función
SELECT_CHREQ() devuelve una lista de los CHREQ con menor nivel
                                level = level_origin_min + 1
                                call SEND_CHREP(id_node_origin, level_origin_min, id_node, level)
                        end for
                else                                    // si sólo se recibe un CHREQ en el intervalo
                        level = level_origin + 1
                        call SEND_CHREP(id_node_origin, level_origin, id_node, level)       // una vez enviado
el CHREP espera un CHACCEPT
                endif
        endif
end while

//    RECEPCIÓN DE UN PAQUETE CHROUTE DE ACTUALIZACIÓN DE TABLA

while CHROUTE recibido do
        if flag_parent = 1 then                         //      si es el ROOT no hace nada
                if level < RT_level then
                // si el nivel del nodo incluido en el mensaje es el mismo que el nivel del que lo recibe, éste no hace
nada (es un vecino y por tanto ya lo meterá en la tabla cuando éste genere un CHREQ
                // si el nivel del nodo que lo recibe es mayor que el incluido en el mensaje, esto sólo puede ocurrir si
alguno de los nodos se ha movido con respecto a su posición inicial en el árbol. No se hace nada
```

```
                // si es menor, el que lo recibe lo incluye en la tabla (inlcuye también a los nodos vecinos del nodo que envía el CHROUTE)
                        RT_QoL = RT_QoL + QoL(id_node, id_node_origin)          // añade al parámetro de calidad total del enlace la parte correspondiente al enlace con su vecino
                        call ADD(RT_node, RT_level, RT_QoL + QoL, id_node_origin, level_origin, NO)

                        call FIND(RT_node, RT_level)
                // busca en su tabla si el nodo guardado ya tenía otra ruta almacenada. En ese caso no reenvía el CHROUTE ya que sería información redundante para sus padres
                        if find = NOT then
                                call CHROUTE(id_node, level, RT_node, RT_level, RT_QoL + QoL)
                        endif
                endif
        endif
end while
}
```

FUNCIONES PRINCIPALES

// ENVÍO DEL PAQUETE CHREQ (es un paquete de broadcast, sin destino concreto) Y ESPERA DE UN CHREP

```
void BROADCAST_CHREQ(id_node_origin,level_origin)
{
enviar_CHREQ(id_node_origin,level_origin)          // llamada a la función que envía directamente el mensaje
num_CHREQ_sent = num_CHREQ_sent + 1                // contador de CHREQ enviados, hasta un máximo de 3

if CHREP recibido then                             // comprueba que el CHREP va dirigido a él
    if (id_node_parent = id_node) and (level_parent = level) and (level_son = level - 1) then
        call SEND_CHACCEPT(id_node_son,level_son,id_node,level)
    endif
else
    if num_CHREQ_sent < 3 then                     // si no recibe un CHREP envía un nuevo CHREQ, hasta un máximo de 3 veces
        call BROADCAST_CHREQ(id_node_origin,level_origin)
    endif
endif
}
```

// FUNCIÓN DE ENVÍO DE CHREP (sí tiene destino concreto, el resto lo ignora) Y ESPERA DE UN CHACCEPT

```
void SEND_CHREP(id_node_parent,level_parent,id_node_son,level_son)
{
enviar_CHREP(id_node_parent,level_parent,id_node_son,level_son)
num_CHREQ_rec = num_CHREQ_rec + 1                  // contador de CHREP enviados, hasta un máximo de 3

if CHACCEPT recibido then                          // comprueba que el CHREP va dirigido a él
    if (id_node_son = id_node) and (level_son = level) then
        call ADD(id_node_parent, level_parent, QoL, , , YES)        // añade el padre a la tabla
```

```
            flag_parent = 1
            call BROADCAST_CHREQ(id_node,level)          // el hijo actúa ahora como padre y envía un CHREQ con su id y su nivel
        endif
else
        if num_CHREP_sent < 3 then          // si no recibe un CHACCEPT envía un nuevo CHREP, hasta un máximo de 3 veces
            call SEND_CHREP(id_node_origin, level_origin, id_node, level)
        endif
endif
}
```

// FUNCIÓN DE ENVÍO DE UN CHACCEPT (sí tiene destino concreto, el resto lo ignora)

```
void SEND_CHACCEPT(id_node_son,level_son,id_node_parent,level_parent)
{
enviar_CHACCEPT(id_node_son,level_son,id_node_parent,level_parent)
num_CHACCEPT_sent = num_CHACCEPT_sent + 1        // contador de CHACCEPT enviados, hasta un máximo de 3

if ACK_entrega_CHACCEPT then                    // si el nivel de enlace da confirmación de entrega del CHACCEPT
        call ADD(id_node_son, level_son, QoL, , , NO)        // añade el hijo a la tabla
        call SEND_CHROUTE(id_node, level, id_node_son, level_son, QoL)        // envía la información con su nuevo hijo hacia arriba
else                            // si el nivel de enlace no da confirmación de entrega del CHACCEPT
        if num_CHACCEPT_sent < 3 then          // si no recibe un CHACCEPT envía un nuevo CHREP, hasta un máximo de 3 veces
            call SEND_CHACCEPT(id_node_son,level_son,id_node,level)
        endif
endif
}
```

FUNCIONES SECUNDARIAS

// FUNCIÓN QUE SELECCIONA EL CHREQ RECIBIDO CON MENOR NIVEL

```
void SELECT_CHREQ()          // devuelve una lista (duplas nodo-nivel) con los nodos de menor nivel
{
level_origin_min = select_min(level_origin)

while level_origin = level_origin_min do
        devolver (id_node_origin,level_origin_min)
end while
}
```

// FUNCIÓN QUE ENVÍA EL PAQUETE DE CONTROL CHROUTE

```
void SEND_CHROUTE(id_node_origin, level_origin, RT_node, RT_level, RT_QoL)
// los parámetros de esta función son los campos de la tabla de rutas más el id_node y level del nodo que lo envía
{
if flag_parent = 2 then                                    // si es el ROOT no hace nada

else
        enviar_CHROUTE(id_node_origin, level_origin, RT_node, RT_level, RT_QoL)
// definimos CHROUTE como un paquete de BROADCAST. El nodo que lo reciba debe comprobar si su información es válida o no para él (lo será si es un padre o un vecino del nodo que lo envía y esa información no está contenida en la tabla). Esto se hace en el programa principal
endif
}
```

// FUNCIÓN QUE ENVÍA EL PAQUETE DE CONTROL CHDEL: mensaje de borrado de rutas que han dejado de ser válidas

```
void SEND_CHDEL(id_node, id_node_parent)
```

```
// la inclusión del padre en la llamada es opcional: si no se incluye se envñia un CHDEl a todos los padres; si se incluye, sólamente se envía a ese padre
// primero selecciona de la tabla las rutas hacia todos sus padres y luego envía un mensaje de borrado a cada uno de ellos
{
if id_node_parent = {vacío} then
        id_node_parent = select (RT_node) where (RT_flag_parent = YES)
        for     id_node_parent do
                enviar_CHDEL(id_node_parent, level_parent, id_node, level)   // level_parent puede eliminarse
        end for
else
        enviar_CHDEL(id_node_parent, level_parent, id_node, level)          // si únicamente se desea borrar la ruta hacia un padre y no la de todos
endif
}

//    FUNCIÓN QUE GUARDA LOS CHREQ RECIBIDOS DURANTE UN INTERVALO DE TIEMPO

void GUARDAR_CHREQ(id_node_origin, level_origin)
{
/*    Cuando se recibe el primer CHREQ se inicia un temporizador (t=0)
        Mientras t < 500 ms (por ejemplo) guarda temporalmente los CHREQ que llegan en ese intervalo
        Cuando t = 500 ms se vuelve al programa principal y se devuelve el nº de CHREQ recibidos (num_CHREQ_rec)
        Puede ocurrir que mientras se esté completando el proceso de asociación con un padre llegue otro CHREQ. En ese caso se ignoraría
*/
}
```

5) Conclusiones y trabajos futuros

En este PFM se ha realizado el diseño teórico de un protocolo jerárquico de encaminamiento salto a salto para una RSI. Este diseño se ha efectuado con base en unas premisas iniciales y según unas necesidades concretas de funcionamiento. El mismo comprende la modularización del protocolo, la definición y diseño de todos los procesos incluidos y una parte de programación básica de los mismos.

Al inicio de este documento se describían las siguientes etapas para el desarrollo de este PFM:

- **Fase 1:** Definición del Proyecto
- **Fase 2:** Planificación del Proyecto
- **Fase 3:** Diseño Conceptual
- **Fase 4:** Codificación y Programación Inicial
- **Fase 5:** Elaboración de la Memoria Final del PFM

Para continuar con la labor realizada podrían establecerse los siguientes hitos y fases que concluirían en una primera etapa cuando el protocolo estuviese totalmente programado, probado y depurado y cuya eficiencia se hubiera comparado en un simulador con la de otros protocolos de encaminamiento existentes en RSI y RAHI.

Una segunda etapa de implementación sería la programación y puesta en marcha del protocolo en una RSI real.

ETAPA 1

- **Fase 6:** Programación del algoritmo para su uso en simulador.

  Uno de los simuladores de redes más extendidos tanto para investigación como para propósitos docentes es el ns-2, el cual tiene algunas capacidades muy interesantes:

- ❖ Simular estructuras y protocolos de redes de todo tipo (satélite, inalámbricas, cableadas, etc.)
- ❖ Desarrollar nuevos protocolos y algoritmos y comprobar su funcionamiento
- ❖ Comparar distintos protocolos en cuanto a prestaciones

Otra alternativa es construir un simulador propio en un lenguaje como *Java* y a continuación programar el algoritmo completo en este lenguaje. El inconveniente de esta tarea es que, si se pretende comparar la eficiencia del protocolo con la de otros más extendidos, sería necesaria también la programación de éstos en el lenguaje del simulador desarrollado.

- **Fase 7:** Pruebas y depuración del algoritmo en el simulador.
- **Fase 8:** Comparación en el simulador de las prestaciones y eficiencia del algoritmo diseñado con otros protocolos de encaminamiento jerárquicos y no jerárquicos. Medida de parámetros y extracción de conclusiones.
- **Fase 9:** Redefinición y cambios en el protocolo para mejorar la eficiencia. Programación y pruebas.

ETAPA 2

- **Fase 10:** Implementación del protocolo en una RSI real: programación del algoritmo en el lenguaje definido para los sensores inalámbricos de la plataforma MICA: nesC.
- **Fase 11:** Pruebas y depuración del algoritmo en un entorno de funcionamiento real.
- **Fase 12:** Medidas de eficiencia del protocolo en una RSI real y extracción de conclusiones.

6) Bibliografía y Referencias

Bibliografía

Wilby, M. R. y Quintela, M. P. *Routing Enquiry Mechanism Utilizing Strategy (REMUS)*

TinyOS, UC Berkeley. http://webs.cs.berkeley.edu/tos, http://www.tinyos.net

Dispositivos sensores inalámbricos MICA. http://www.xbow.com

Baccelli, F., Kofman, D. y Rougier, J. L. (1999). *Self organizing hierarchical multicast trees and their optimization.* Proc. of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1081 – 1089.

Caleffi, M., Ferraiuolo, G. y Paura, L. (2007). *Augmented Tree-based Routing Protocol for Scalable Ad Hoc Networks.* IEEE International Conference on Mobile Ad hoc and Sensor Systems, pp. 1-6.

Thepvilojanapong, N., Tobe, Y. y Sezaki, K. (2005). *HAR: Hierarchy-Based Anycast Routing Protocol for Wireless Sensor Networks.* Proc. of the 2005 Symposium on Applications and the Internet.

Ko, Y.-B. y Vaidya, N. H. (1998). *Location-aided routing (LAR) in mobile ad hoc networks.* Proc. of International Conference on Mobile Computing and Networking, pp. 66–75.

Karp, B. y Kung, H. T. (2000). *GPSR: greedy perimeter stateless routing for wireless networks.* Proc. of International Conference on Mobile Computing and Networking, pp. 243–254,.

Eriksson, J., Faloutsos, M. y Krishnamurthy, S. (2007). *DART: Dynamic Address Routing for Scalable Ad Hoc and Mesh Networks.* IEEEACM Transactions on Networking, vol.15, nº 1, pp. 119-132.

Abolhasan, M., Wysocki, T. y Dutkiewicz, E. (2004). *A review of routing protocols for mobile ad hoc networks*. Ad Hoc Networks, vol 2, nº 1, pp. 1–22.

Broch, J. et al. (1998). *A performance comparison of multi-hop wireless ad hoc network routing protocols*. Proc. of MOBICOM, pp. 85–97.

Referencias

[1] http://www.ambientintelligence.org

[2] Corson, S. y Macker, J. (1999). *Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations*. RFC 2501 memo de la IETF Network Working Group.

[3] IEEE std 802.11-1997 *information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: Wireless LAN Medicum Access Control (MAC) and Physical layer (PHY) specifications*. Technical report.

[4] Karn, P. (1990). *MACA - A New Channel Access Method for Packet Radio*. ARRL Computer NeConferencetworking, London, Ontario, Canada.

[5] Talucci, F. y Gerla, M. (1997). *A wireless MAC protocol for high speed ad hoc networking*, Universal Personal Communications Record.

[6] Garcés, R. y García-Luna-Acebes, J. J. (1996). *Floor acquisition multiple access with collision resolution*. Proc. of the 2nd International Conference on Mobile Computing and Networking, pp. 187-197.

[7] Holland, G. D. (2004). *Adaptive models for Mobile Ad Hoc Networks*. Department of Computer Science. College Station, TX: Texas A&M University.

[8] Anastasi, G., Ancillotti, E., Conti, M. y Passarella, A. (2005). *TPA: a transport protocol for ad hoc networks*. Proc. of the 10th IEEE Symposium on Computers and Communications, pp. 51-56.

[9] Zjalic, D., Pavlovic, M. y Aralica, M. (2004). *Stream control transport protocol (SCTP).* Proc. of the 46th International Symposium Electronics in Marine, pp. 160-165.

[10] Shiwen Mao, Bushmitch, D., Narayanan, S. y Panwar, S.S. (2006). *MRTP: a multiflow real-time transport protocol for ad hoc networks*. IEEE Transactions on Multimedia, vol. 8, nº 2, pp. 356 - 369

[11] Basagni, S., Conti, M., Giordano, S. y Stojmenovic, I. (2004). *Mobile Ad Hoc Networking*, capítulo 10, pp. 275-300. Wiley-IEEE Press.

[12] Ramanathan, S. y Steenstrup, M. (1996). *A survey of routing techniques for mobile communications networks*. Mobile Networks and Applications, vol. 1, nº 2, pp. 89–104.

[13] Royer, E. M. y Toh, C. K. (1999). *A review of current routing protocols for ad-hoc mobile wireless networks*. IEEE Personal Communications, vol. 6, nº 2, pp. 46–55.

[14] Feeney, L. M. (1999). *A taxonomy for routing protocols in mobile ad hoc networks*. Sics technical report T99/07.

[15] Liu, C. y Kaiser, J. (2003). *A survey of mobile ad hoc network routing protocols*. Technical report tr-4. nr. 2003-08, Universidad de Ulm.

[16] Lang, D. (2003). *A comprehensive overview about selected ad-hoc networking routing protocols*. Technical report. Department of Computer Science, Universidad Técnica de Munich.

[17] Vinagre, J. J. (2007). *Teoría del Encaminamiento en Redes Ad Hoc Inalámbricas*. Tesis doctoral. Universidad Carlos III de Madrid. Cap. 1, pp. 4-7.

[18] Kulik, J., Heinzelman, W. R. y Balakrishnan, H. (2002). *Negotiation-based protocols for disseminating information in wireless sensor networks*. Wireless Networks, vol. 8, pp. 169-185.

[19] Intanagonwiwat, C., Govidan, T. y Estrin, D. (2000). *Direct Diffusion: a scalable and robust communication paradigm for Networks*. Proc. of ACM MobiCom '00, Boston, MA, pp. 56-57.

[20] Braginsky, D. y Estrin, D. (2002). *Rumor Routing Algorithm for Sensor Networks*. Proc. of the 1st Workshop on Sensor Networks and Applications, Atlanta, GA.

[21] Ye, F., Chen, A., Liu, S. y Zhang, L. (2001). *A scalable solution to minimum cost forwarding in large sensor networks*. Proc. of the 10th International Conference on Computer Communications and Networks, pp. 304-309.

[22] Schurgers, C. y Srivastava, M. B. (2001). *Energy efficient routing in wireless sensor networks*, MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force, McLean, VA.

[23] Heinzelman, W., Chandrakasa, A. y Balakrishnan, H. (2000). *Energy-Efficient Communication Protocol for Wireless Microsensor Networks*, Proc. of the 33rd Hawaii International Conference on System Sciences.

[24] Lindsey, S. y Raghavendra, C. (2002). *PEGASIS: Power-Efficient Gathering in Sensor Information Systems"*. IEEE Aerospace Conference Proceedings, vol. 3, pp. 1125-1130.

[25] Manjeshwar, A. y Agarwal, D. P. (2001). *TEEN: a routing protocol for enhanced efficiency in wireless sensor networks*. 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing.

[26] Manjeshwar, A. y Agarwal, D. P. (2002). *APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks.* Parallel and Distributed Processing Symposium, pp. 195-202.

[27] Rodoplu, V. y Meng, T. H. (1999). *Minimum Energy Mobile Wireless Networks*, IEEE Journal Selected Areas in Communications, vol. 17, nº 8, pp. 133344.

[28] Ye, F., Luo, H., Cheng, J., Lu, S. y Zhang, L. (2002). *A Two-tier data dissemination model for large-scale wireless sensor networks.* Proc. of ACM/IEEE MOBICOM.

[29] Xu, Y., Heidemann, J. y Estrin, D. (2001). *Geography-informed Energy Conservation for Ad-hoc Routing*. Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 70-84.

[30] Yu, Y., Estrin, D. y Govindan, R. (2001). *Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks*, UCLA-CSD TR-01-0023.

[31] Chang, J. H. y Tassiulas, L. (2000). *Maximum Lifetime Routing in Wireless Sensor Networks*. Proc. of the Advanced Telecommunications and Information Distribution Research Program, College Park, MD.

[32] Dulman, S., Nieberg, T., Wu, J. y Havinga, P. (2003). *Trade-Off between Traffic Overhead and Reliability in Multipath Routing for Wireless Sensor Networks*. WCNC Workshop, New Orleans, Lousiana, EE. UU.

[33] Sohrabi, K. y Pottie, J. (2000). *Protocols for self-organization of a wireless sensor network.* IEEE Personal Communications, vol. 7, nº 5, pp. 16-27.

[34] He, T. et al. (2003). *SPEED: A stateless protocol for real-time communication in sensor networks.* Proceedings of International Conference on Distributed Computing Systems, Providence, RI.

7) Apéndices

7.1.- HOJA DE CARACTERÍSTICAS del dispositivo sensor inalámbrico MICAz

New

MICAz

WIRELESS MEASUREMENT SYSTEM

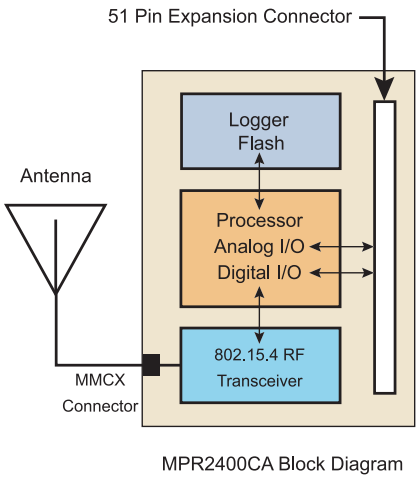
- IEEE 802.15.4, Tiny, Wireless Measurement System
- Designed Specifically for Deeply Embedded Sensor Networks
- 250 kbps, High Data Rate Radio
- Wireless Communications with Every Node as Router Capability
- Expansion Connector for Light, Temperature, RH, Barometric Pressure, Acceleration/Seismic, Acoustic, Magnetic, and other Crossbow Sensor Boards

Applications

- Indoor Building Monitoring and Security
- Acoustic, Video, Vibration, and Other High Speed Sensor Data
- Large Scale Sensor Networks (1000+ Points)
- ZigBee Compliant Systems and Sensors

ZigBee™ Alliance

A member of the ZigBee Alliance



MPR2400CA Block Diagram

MICAz

The MICAz is a 2.4GHz, IEEE 802.15.4 compliant, Mote module used for enabling low-power, wireless, sensor networks. The MICAz Mote features several new capabilities that enhance the overall functionality of Crossbow's MICA family of wireless sensor networking products. These features include:

- IEEE 802.15.4/ZigBee compliant RF transceiver
- 2.4 to 2.4835 GHz, a globally compatible ISM band
- Direct sequence spread spectrum radio which is resistant to RF interference and provides inherent data security
- 250 kbps data rate
- Runs TinyOS 1.1.7 and higher, including Crossbow's reliable mesh networking stack software modules
- Plug and play with all of Crossbow's sensor boards, data acquisition boards, gateways, and software

TinyOS is a small, open-source, energy-efficient, software operating system developed by UC Berkeley which supports large scale, self-configuring sensor networks. The source code software development tools are publicly available at:

http://webs.cs.berkeley.edu/tos

Processor and Radio Platform (MPR2400CA):
Using TOS, a single processor board (MPR2400CA) can be configured to run your sensor application/processing and the mesh networking radio stack simultaneously. The MICAz IEEE 802.15.4 radio offers both high speed (250 kbps) and hardware security (AES-128). The MICAz 51-pin expansion connector supports Analog Inputs, Digital I/O, I2C, SPI, and UART interfaces. These interfaces make it easy to connect to a wide variety of external peripherals.

Sensor Boards:
In addition to the MTS101 and MTS300/310 series, Crossbow offers a variety of sensor and data acquisition boards for the MICAz Mote. All of these boards connect to the MICAz via the standard 51-pin expansion connector. Custom sensor and data acquisition boards are also available. Please contact Crossbow for additional information.

wireless sensor networks

Document Part Number: 6020-0060-01 Rev A

Processor/Radio Board	MPR2400CA	Remarks
Processor Performance		
Program Flash Memory	128K bytes	
Measurement Serial Flash	512K bytes	>100,000 measurements
Configuration EEPROM	4 K bytes	
Serial Communications	UART	0 V to 3 V transmission levels
Analog to Digital Converter	10 bit ADC	8 channels, 0 V to 3 V input
Other Interfaces	Digital I/O,I2C,SPI	
Current Draw	8 mA	Active mode
	< 15uA	Sleep mode
RF Transceiver		
Frequency band[1]	2400 MHz to 2483.5 MHz	ISM band, programmable in 1 MHz steps
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	1/2 wave dipole antenna, LOS
Indoor Range	20 m to 30 m	1/2 wave dipole antenna
Current Draw	19.7 mA	Receive mode
	11 mA	TX, -10 dBm
	14 uA	TX, -5 dBm
	17.4 mA	TX, 0 dBm
	20 µA	Idle mode, voltage regular on
	1 µA	Sleep mode, voltage regulator off
Electromechanical		
Battery	2X AA batteries	Attached pack
External Power	2.7 V - 3.3 V	Molex connector provided
User Interface	3 LEDs	Red, green, and yellow
Size (in)	2.25 x 1.25 x 0.25	Excluding battery pack
(mm)	58 x 32 x 7	Excluding battery pack
Weight (oz)	0.7	Excluding batteries
(grams)	18	Excluding batteries
Expansion Connector	51 pin	All major I/O signals

Notes

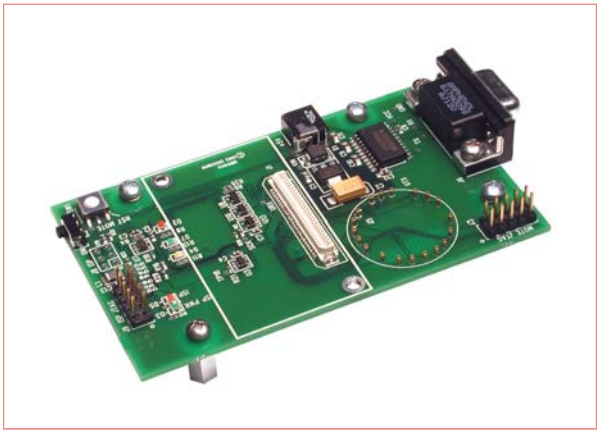
[1] 5 MHz steps for compliance with IEEE 802.15.4/D18-2003.

Specifications subject to change without notice

Base Stations:

A base station allows the aggregation of sensor network data onto a PC or other computer platform. Any MICAz Mote can function as a base station by plugging the MPR2400CA Processor/Radio Board into an MIB510CA serial interface board. The MIB510CA provides an RS-232 serial interface for both programming and data communications. Crossbow also offers a stand-alone gateway solution, the MIB600CA, for TCP/IP-based Ethernet networks.

▼ MIB510CA Mote Interface Board



Ordering Information

Model	Description
MPR2400CA	2.4 GHz Processor/Radio Board
MTS101CA	Light, Temp, and Prototype Sensor Board
MTS300CA	Light, Temp, Acoustic, and Sounder Sensor Board
MTS310CA	Same as MTS300CA but also includes Magnetic and Acceleration
MIB510CA	MICA, MICA2, MICA2DOT Mote Interface & Programming Board

Document Part Number: 6020-0060-01 Rev A