Universidad de Oriente

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones

TRABAJO DE DIPLOMA

Algoritmos de enrutamiento inteligentes para redes de sensores inalámbricos

Autor: Elier Alejandro Carcasés Bonilla

Tutor: Dra.C. María Margarita Goire Castilla

M.Sc. Lídice Romero Amondaray.

Santiago de Cuba

Junio, 2015

Universidad de Oriente

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones

TRABAJO DE DIPLOMA

Algoritmos de enrutamiento inteligentes para redes de sensores inalámbricos

Autor: Elier Alejandro Carcasés Bonilla

elier.carcases@tle.fie.uo.edu.cu

Tutor: Dra.C. María Margarita Goire Castilla

Profesora Titular, Departamento de Telecomunicaciones, Facultad de Ingeniería Eléctrica

mgoire@fie.uo.edu.cu

M.Sc. Lídice Romero Amondaray

Profesora Auxiliar, Departamento de Telecomunicaciones, Facultad de Ingeniería Eléctrica,

lidice@fie.uo.edu.cu

Santiago de Cuba

Junio, 2015

COMPROMISO DEL AUTOR

Hago constar que el presente trabajo de diploma es de mi autoría exclusivamente, no constituyendo copia de ningún trabajo realizado anteriormente y las fuentes usadas para la realización del trabajo se encuentran referidas en la bibliografía. Doy mi consentimiento a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización del Tutor o Institución.

_____

Firma del Autor

PENSAMIENTO

"Si buscas resultados distintos no hagas siempre lo mismo".

Albert Einstein

DEDICATORIA

A mi abuela Onelia y mi tía Olivia, mis primeras maestras, ya tienen un ingeniero.

A mis padres, porque gracias a ellos mi único deber es el estudio.

AGRADECIMIENTOS

A mis abuelas Onelia y Olivia, por estar siempre pendientes de mi progreso como estudiante.

A mis padres, por su compresión y apoyo.

A mis amigos, que me dieron aliento, en especial Darío, Sonia y Otto, por ayudarme incondicionalmente cuando más lo necesitaba, animarme, y estar presente en los momentos más duros.

A mis tutoras, por su apoyo y orientación durante todo el período de investigación.

A mis compañeros de aula: Humbertico, Manuel, Bernardino, Albertico, David, Hassan, por los momentos compartidos durante estos cinco años de carrera.

A Carlitos, Omar, y a todos aquellos que de una forma u otra contribuyeron a que este trabajo fuese posible.

Al colectivo de profesores del Departamento de Telecomunicaciones y Electrónica, por haberme formado como profesional.

RESUMEN

Uno de los parámetros críticos en el diseño de una red de sensores inalámbricos es el tiempo de duración de la batería de los nodos sensores. Este aspecto está vinculado al enrutamiento producto de que los nodos consumen más energía en las tareas de comunicación. Este trabajo estuvo dedicado al estudio de los algoritmos de enrutamiento, especialmente a los inteligentes, y particularmente a los conscientes de la energía. Se identificó que los algoritmos inteligentes basados en aprendizaje por refuerzos eran los adecuados para implementar protocolos de enrutamiento en estas redes, debido a sus bajas demandas de cómputo y memoria, y de ser totalmente distribuidos. Se estudió el protocolo inteligente FROMS, el cual provee un esquema de encaminamiento consciente de la energía, y se implementó en el simulador Castalia. Se introdujeron nuevas funcionalidades al simulador, al dotarlo del protocolo FROMS que no estaba implementado, lo que exigió modificar el módulo de administración de recursos de Castalia. Se realizó una comparación de FROMS con el protocolo convencional REL, tomando como criterios el tiempo de vida de la red y el manejo de la energía de los nodos. Los resultados mostraron que FROMS, en el escenario que se simuló, puede extender el tiempo de vida de la red en un 17 % en comparación con REL.

Palabras clave: sensores inalámbricos, enrutamiento, algoritmos inteligentes, energía, FROMS

ABSTRACT

One of the critical parameters in wireless sensor networks design is the battery lifetime of the sensor nodes. This aspect is linked to routing because nodes consume more energy when performing communication tasks. This project was addressed to the study of routing algorithms, especially the intelligent ones, and particularly to energy-aware algorithms. It was identified that reinforcement learning based algorithms are the most suitable for implementing routing protocols for this kind of networks, due to their low demands of processing and memory, and being fully distributed. The intelligent routing protocol FROMS was studied, which provides an energy-aware routing scheme, and it was implemented in Castalia. New functionalities were inserted into the simulator, by equipping it with FROMS protocol, which was not implemented, task that demanded some modifications into Castalia's resource manager module. A comparison between FROMS and conventional protocol REL was made, taking for criteria the network lifetime and management of the node's energy. Results showed that FROMS, in the simulated scenario, can extend the network lifetime by 17 % in comparison to REL.

Keywords: *wireless sensors, routing, intelligent algorithms, energy, FROMS*

ÍNDICE

INTRODUCCIÓN

Una red de sensores inalámbricos (WSN por sus siglas en inglés) es una red de diminutos dispositivos equipados con sensores que colaboran en una tarea común y están distribuidos en un área geográfica determinada. Estos dispositivos, conocidos como nodos sensores o motas, son unidades autónomas capaces de medir determinadas condiciones físico-ambientales en el entorno, realizar algún tipo de procesamiento sobre los datos, y encaminarlos hacia una estación base [1].

Los principales tipos de aplicación de las redes de sensores inalámbricos son: reporte periódico, detección de eventos y almacenamiento de datos. En este sentido, es importante destacar que las principales características de cada WSN quedan determinadas por los requerimientos de la aplicación para la cual se desarrolla.

Estas redes soportan topologías en árbol de *clusters*, estrella y malla, y permiten formar redes *ad-hoc* sin infraestructura física preestablecida ni administración central. Los nodos sensores tienen una limitada potencia de transmisión y restricciones en el ancho de banda disponible. Por lo tanto, es necesario que colaboren entre ellos mediante un esquema multi-salto para conseguir un objetivo común: que cualquier paquete llegue a su destino aunque éste no sea accesible directamente desde el origen.

Las WSN tienen capacidad de auto restauración, es decir, si se avería un nodo, los demás encontrarán nuevas vías para encaminar los paquetes de datos. De esta forma, sobrevivirá en su conjunto, aunque algunos nodos hayan caído por falta de energía o roturas. Esta capacidad de adaptación es consecuencia de los protocolos de enrutamiento utilizados. La elección del protocolo depende de la aplicación y las características propias de la red [2].

Un algoritmo o protocolo de enrutamiento determina el camino de un mensaje desde una fuente hacia un destino. Este camino se debe pensar en base a un objetivo planeado, que puede ser máximo tiempo de vida de la red, seguridad de que todos los mensajes lleguen al nodo sumidero, mínima sobrecarga de la red, entre otros. Un buen camino es aquel que proporciona costo mínimo de acuerdo al objetivo planteado, no siempre el camino más corto es el más conveniente.

Toda técnica de encaminamiento propuesta para trabajar en una WSN debe ser eficiente en cuanto al gasto de energía, ya que generalmente los nodos están alimentados a baterías, y habitualmente se encuentran en lugares donde es difícil su reposición, de manera que se hace conveniente tomar medidas que contribuyan a maximizar en lo posible la vida de la red [2].

Las WSN heredan algunos de los protocolos típicos de redes *ad hoc* como AODV (*Ad hoc On Demand Distance Vector*), que es el utilizado por defecto en redes ZigBee de topología mallada. Dentro de este conjunto se encuentra además DSR (*Dynamic Source Routing*), que se puede usar en redes compuestas por nodos móviles. Estos protocolos se adaptan bien a este tipo de redes debido a su capacidad para encontrar nuevas rutas y adaptarse a cambios en la topología de la red. Sin embargo, cuando el número de nodos es elevado, no escalan bien, ya que conllevan almacenar y gestionar largas tablas caché de rutas. Además, no se ajustan a los requerimientos de las WSN, fundamentalmente en lo que respecta a energía, cómputo y comunicación [3].

El creciente número de aplicaciones para las WSN, y especialmente sus requerimientos heterogéneos y propiedades, demandan nuevos protocolos de comunicación. Las motas tienen limitaciones de hardware y de energía disponible, y se ha determinado que los procesos de comunicación son los que consumen la mayor parte de la energía de las baterías, factor que es decisivo en el tiempo de vida de la red [4].

Es por ello que el desarrollo de protocolos de enrutamiento eficientes en este sentido es una de las áreas de investigación con más movimiento en el ámbito de las WSN [5]. LEACH (*Low Energy Adaptive Clustering Hierarchy*) [6] y los protocolos de la familia SPIN (*Sensor Protocol for Information Via Negotiation*) [7] son algunos de los protocolos diseñados específicamente para extender el tiempo de vida de una red de sensores inalámbricos. Como medida de optimización buscan adaptar el nivel de participación de cada nodo en las labores de encaminamiento en base a su energía residual.

A pesar de la inmensa cantidad de propuestas, problemas fundamentales quedan por resolver, principalmente en cuanto a la eficiencia energética para diferentes escenarios de aplicación. Dadas las características y complejidad de las WSN, resulta difícil encontrar un algoritmo de enrutamiento que funcione bien para la mayoría de los casos. Es una tecnología relativamente nueva donde el rol más importante de los nodos es su grado de participación y responsabilidad en la organización y gestión de la red, además del cómputo

que ejecutan para lograrlo. Esto sugiere proveer a los nodos de cierto nivel de inteligencia; los nodos serán más o menos inteligentes en las decisiones si se los provee de los datos correctos y del tipo y calidad de cómputo que ejecuten [8].

La Inteligencia Computacional (IC) se presenta como una solución para resolver problemas de optimización relacionados con el enrutamiento en redes de sensores. IC es el estudio de mecanismos adaptativos que permiten o facilitan el comportamiento inteligente en ambientes complejos y cambiantes. Estos mecanismos incluyen paradigmas que exhiben una habilidad de aprender o adaptarse a nuevas situaciones, generalizar, descubrir y asociar.

IC envuelve paradigmas tales como redes neuronales, aprendizaje por refuerzos, inteligencia de enjambres, algoritmos evolutivos, lógica difusa y sistemas inmunes artificiales. Permite un comportamiento autónomo y generalmente es robusto ante cambios de topologías y fallos en la comunicación [9].

Los algoritmos de enrutamiento inteligentes están basados generalmente en redes neuronales, inteligencia de enjambres o aprendizaje por refuerzos. Sin embargo, los paradigmas de IC subyacentes tienen diferentes requerimientos computacionales y criterios de convergencia. Estas propiedades deben ser consideradas dependiendo de la aplicación, principalmente en cuanto al tráfico generado, flexibilidad, robustez ante cambios de topología y movilidad de los nodos.

Por ejemplo, los algoritmos de enrutamiento basados en inteligencia de enjambres (específicamente en colonias de hormigas) son muy flexibles, pero generan gran cantidad de tráfico. En un escenario de WSN esta carga debe ser cuidadosamente considerada.

Por otra parte, algoritmos que requieren una fase de aprendizaje *off-line*, como los algoritmos genéticos o las redes neuronales, no pueden adaptarse en redes cuya topología está en constante cambio. Tampoco proveen un esquema de encaminamiento consciente de la energía. Requieren altos costos de reunir previamente los datos relevantes en la estación base para luego calcular el árbol de enrutamiento y diseminar la información de encaminamiento a los nodos. Los fallos en la comunicación no pueden ser considerados, y en caso de un cambio en la topología de la red, todo el procedimiento debe iniciarse nuevamente [9].

Los algoritmos basados en aprendizaje por refuerzos [10] son muy efectivos y pueden ser implementados con bajos requerimientos de cómputo y memoria, en comparación con los otros paradigmas. Esta última alternativa es la más utilizada para la resolución de problemas de optimización relacionados con el enrutamiento, por su flexibilidad, carácter distribuido, y ser muy robusto ante cambios de topología [11].

Usualmente, los desarrolladores de técnicas para WSN no son conscientes de las potencialidades que las técnicas de Inteligencia Computacional ofrecen. Por el otro lado, los investigadores de técnicas de Inteligencia Computacional no están familiarizados con los problemas reales y los requerimientos de las WSN. Esta desigualdad dificulta mucho la colaboración y el desarrollo. Numerosas comunidades de investigación en el ámbito de la Inteligencia Computacional desarrollan propuestas concurrentemente, por lo que no existe una visión general, y la mayoría de los trabajos están dispersos en revistas y conferencias cuyo enfoque principal no son las WSN [9].

A pesar de la multitud de algoritmos y protocolos propuestos, muy pocos protocolos han crecido fuera del ambiente de simulación. La mayoría de ellos no consideran enlaces no confiables o asimétricos, fallas en los nodos y movilidad. Además, un problema común es la falta de comparación con protocolos convencionales para identificar claramente las ventajas de la introducción de técnicas de Inteligencia Computacional [9].

Con frecuencia las simulaciones no se realizan sobre simuladores de redes o en bancos de prueba reales con motes programables. En estos casos, se eligen programas como MATLAB o alguna aplicación desarrollada específicamente para la simulación del algoritmo, en lugar de simuladores como NS-2, OMNeT++, o Castalia. Esto impide evaluar el comportamiento real del algoritmo de enrutamiento sobre un protocolo MAC (*Medium Access Control*) real.

Problema de la investigación

¿Cómo podría contribuir la Inteligencia Computacional al enrutamiento en las redes de sensores inalámbricos?

Objeto de estudio

Algoritmos de enrutamiento para redes de sensores inalámbricos.

Campo de acción

Algoritmos de enrutamiento basados en Inteligencia Computacional.

Objetivo general

Evaluar el desempeño de un algoritmo de enrutamiento inteligente en una red de sensores inalámbricos tomando como criterio el tiempo de vida de la red.

Objetivos específicos

- Analizar la información relacionada con los algoritmos de enrutamiento convencionales y las técnicas de Inteligencia Computacional aplicadas al enrutamiento en redes de sensores.
- Estudiar las características de los simuladores para redes de sensores inalámbricos.
- Implementar uno de los algoritmos de enrutamiento usando técnicas inteligentes.
- Simular un protocolo de enrutamiento inteligente para redes de sensores inalámbricos.
- Comparar el algoritmo implementado con un algoritmo convencional similar en cuanto a la métrica utilizada.

Tareas de la investigación

1. Investigación de los algoritmos de enrutamiento convencionales e inteligentes para seleccionar los que serán objeto de estudio en cada caso.
2. Investigación de las herramientas de simulación de redes computadoras que permitan implementar algoritmos de enrutamiento.
3. Codificación de los algoritmos de enrutamiento seleccionados.
4. Simulación de los algoritmos de enrutamiento seleccionados.
5. Evaluación del desempeño de los algoritmos simulados para poder realizar las comparaciones de acuerdo a las métricas consideradas y comprobar las potencialidades de la Inteligencia Computacional.
6. Redacción del informe.

Hipótesis

Si se usan algoritmos de enrutamiento inteligentes en redes de sensores inalámbricos, entonces se puede prolongar el tiempo de vida de la red.

CAPÍTULO 1. PROTOCOLOS DE ENRUTAMIENTO PARA REDES DE SENSORES INALÁMBRICOS

Las redes de sensores inalámbricos por lo general son multi-salto para enfrentar las fuertes limitaciones de potencia de transmisión. En este esquema los nodos intermedios en la ruta actúan como encaminadores, ahorrando la necesidad de dispositivos adicionales. Existen múltiples protocolos que facilitan este tipo de comunicación. La elección del mismo varía de acuerdo a las características de la topología de red y requerimientos específicos de la aplicación [12].

En este capítulo se abordan las características del enrutamiento en redes de sensores. Se explica la necesidad del ahorro de energía, y se presentan algunos de los protocolos de enrutamiento tradicionales, y los basados en Inteligencia Computacional. Finalmente, se realiza una comparación entre los diferentes paradigmas de Inteligencia Computacional y se presentan algunos de los simuladores para redes de sensores inalámbricos.

1.1 Particularidades del enrutamiento

Un algoritmo de enrutamiento determina el camino de un mensaje desde una fuente hacia un destino. Generalmente dicho camino está compuesto por varios tramos que unen distintos nodos entre sí. Cada nodo envía sus mensajes al siguiente en la ruta, y así sucesivamente, hasta que el mensaje llegue al destino. Este camino se debe pensar sobre la base de un objetivo planeado, que puede ser maximizar el tiempo de vida de la red, mínima sobrecarga, o seguridad de que todos los mensajes lleguen al nodo *sink*, entre otros. Un buen camino es aquel que proporciona costo mínimo de acuerdo al objetivo planteado, no siempre el camino más corto es el más conveniente.

Generalmente los nodos no tienen previo conocimiento de la topología de la red, por lo que deben realizar acciones de descubrimiento de rutas. La idea básica es que cuando un nodo se une a la red, anuncia su presencia y escucha los anuncios *broadcast* de sus vecinos. El nodo se informa del estado de sus vecinos y de la manera de encaminar paquetes a través de ellos. A su vez, puede anunciar al resto de nodos que pueden ser accedidos desde él.

Transcurrido un tiempo, cada nodo sabrá qué nodos tiene alrededor, y una o más formas de alcanzarlos [13].

Los protocolos de enrutamiento para redes de sensores tienen diferentes clasificaciones de acuerdo a su esquema de direccionamiento. Existen protocolos centrados en los datos, generalmente usados en redes con una alta densidad de nodos. En este tipo de protocolos, lo que se direcciona no es un nodo en sí, sino el dato que se desea encaminar, a través de un identificador o meta-dato. De esta manera se reduce el tráfico redundante generado por nodos que cubren una misma área geográfica.

También existen protocolos jerárquicos, que dividen la red en dos o más niveles. Los nodos dentro de un mismo nivel encaminan los datos hacia un nodo con mayor capacidad de batería, denominado cabeza de *cluster*. Este nodo es capaz de reducir el tráfico mediante la fusión de los datos recibidos, a la vez que libra a los demás nodos de encaminar los datos de sus vecinos.

La última clasificación se corresponde a los protocolos de enrutamiento para redes planas. Se utilizan en redes donde todos los nodos cumplen la misma función, es decir, las motas pueden realizar mediciones de las variables ambientales, y también pueden encaminar los datos recibidos de nodos vecinos.

A pesar de las diferentes clasificaciones, todos los protocolos propuestos para trabajar en una WSN deben cumplir con las siguientes condiciones:

- Se debe lograr un balance adecuado entre los beneficios del *overhead* y la disminución causada en la eficiencia de la red por la transmisión de datos de control.
- Mantener una tabla de enrutamiento razonablemente pequeña.
- Elegir la mejor ruta para un destino dado (ya sea la más rápida, confiable, de mejor capacidad, o de menor coste).
- Mantener la tabla regularmente para actualizarse ante la caída de nodos, cambios de posición o aparición de nuevos vecinos.
- Requerir una pequeña cantidad de mensajes y tiempo para converger [13].

1.1.1 Consciencia de la energía

La energía es utilizada por los nodos en tres funciones principales: medición, procesamiento y comunicación. Se ha comprobado que el proceso de comunicación consume la mayor

parte de la energía disponible, así que se intenta minimizar la tarea de diseminación realizando todo el procesamiento local posible [13].

Una de las limitaciones fundamentales de las redes de sensores es la capacidad de batería. En las redes móviles y ad hoc los protocolos se diseñan con el objetivo de minimizar la cantidad de saltos que atraviesa un paquete. A diferencia de este tipo de redes, en las WSN es crucial minimizar el consumo de energía empleado para la tarea de encaminamiento, o resolver el problema del encaminamiento de menor energía a partir de diferentes enfoques de diseño, que se manifiestan en las métricas de energía utilizadas.

La técnica de dar consciencia de la energía al encaminamiento no aparece como un paradigma en sí misma, sino que la eficiencia en términos de energía se busca utilizando técnicas adicionales. El empleo de métricas de energía en el encaminamiento no siempre aparece en el diseño de protocolos que buscan eficiencia en el uso de la energía [13].

En [14] se presentan varias métricas de diseño que permiten construir rutas eficientes en términos de energía, y que aparecen muy frecuentemente en los protocolos para redes de sensores:

- Minimizar la energía total de la ruta

Con el enfoque de energía total mínima o MTE (*Minimum Total Energy*), se busca minimizar la energía de transmisión y recepción total consumida por un paquete para alcanzar el destino. Sucede que si todo el tráfico se encamina a través de las rutas más económicas, los nodos en esa ruta se agotan rápidamente, particionando la red, por lo que no se recomienda utilizar esta métrica en forma aislada.

- Maximizar el tiempo hasta que la red se particiona.

Se determinan los nodos que al ser extraídos causan particiones, y se trata de balancear la carga sobre los mismos.

- Uniformizar el consumo de energía

Se parte de la hipótesis de que todos los nodos son igualmente importantes, y se intenta asegurar que todos los nodos funcionen juntos la mayor cantidad de tiempo mediante la distribución de la carga de enrutamiento entre ellos. Este objetivo se puede alcanzar mediante el uso de rutas compuestas por los nodos de mayor energía residual, evitando los nodos de menor capacidad restante. Se sacrifica el uso de rutas más cortas por el beneficio de posponer al máximo posible la muerte del primer nodo.

- Minimizar el costo por paquete

Se seleccionan las rutas de menor costo total, que equivale a la suma de los costos de cada enlace que se utiliza. Si se define adecuadamente el costo del enlace, puede utilizarse la métrica con diferentes objetivos. Por ejemplo, si el costo depende de la energía residual, se puede lograr aumentar el tiempo hasta el particionamiento de la red [14].

1.2 Protocolos de enrutamiento tradicionales

Debido a la heterogeneidad de los escenarios de aplicación para las redes de sensores inalámbricos, existe una gran cantidad de protocolos de enrutamiento diseñados para cubrir las necesidades de cada una de ellas. En los siguientes apartados se describen algunos de los protocolos de enrutamiento más populares en redes de sensores, atendiendo a las diferentes clasificaciones.

1.2.1 AODV

AODV es el protocolo usado por defecto en redes ZigBee de topología mallada, y pertenece al conjunto de protocolos de enrutamiento para redes planas. Fue diseñado para adaptarse rápidamente a cambios en la topología de la red, con bajas demandas de procesamiento y comunicación.

Cada destino dentro de la tabla de enrutamiento tiene asociado un temporizador y un número de secuencia. La función del temporizador es evitar el uso de enlaces de los que no se ha recibido reportes de estado desde hace mucho tiempo, mientras que el número de secuencia asociado permite distinguir entre información nueva e información antigua. De esta manera se puede evitar la formación de bucles y la transmisión de rutas obsoletas.

El descubrimiento de rutas está basado en ciclos de petición y respuesta, y la métrica utilizada es el número de saltos desde la fuente hasta el destino. Cuando un nodo desea enviar datos a un destino envía mediante *broadcast* un mensaje de petición de rutas RREQ (*Route Request*). Este mensaje se propaga a través de la red, y cualquier nodo que conozca una ruta hacia el destino, o el mismo destino, contestarán a la petición con la ruta solicitada mediante un mensaje RREP (*Route Reply*) [15].

En la figura 1.1 se muestra el formato del paquete de solicitud de ruta. Contiene las direcciones de red de origen y destino. También contiene un ID de solicitud, que es un contador local que se mantiene por separado en cada nodo y se incrementa cada vez que se difunde un paquete de solicitud de ruta. En conjunto, los campos dirección de origen e

id de solicitud identifican de manera única al paquete de solicitud de ruta, a fin de que los nodos descarten cualquier duplicado que pudieran recibir.

Dirección de origen	ID de solicitud	Dirección de destino	# de secuencia de origen	# de secuencia de destino	Cuenta de saltos

Figura 1.1. Formato del paquete solicitud de ruta (Fuente: [15]).

Además del contador ID de solicitud, cada nodo mantiene también un segundo contador de secuencia que se incrementa cada vez que se envía un paquete de solicitud de ruta (o una respuesta al paquete de solicitud de ruta de algún otro nodo). El cuarto campo es un contador de secuencia del emisor del mensaje de descubrimiento, y el quinto campo representa el valor más reciente del número de secuencia que el origen ha recibido del destino (0 si nunca se han comunicado). El último campo es un contador de saltos, se inicializa en 0 y se incrementa con cada salto que realiza el paquete.

Cuando un paquete de solicitud de ruta llega a un nodo vecino se procesa de la siguiente manera:

1. El par (Dirección de origen, ID de solicitud) se busca en una tabla de historia local para ver si esta solicitud ya se había recibido y procesado. Si es un duplicado, se descarta y el procesamiento se detiene. En caso contrario, el par se introduce en la tabla de historia a fin de que se puedan rechazar futuros duplicados, y el procesamiento continúa.

2. El receptor busca el destino en su tabla de enrutamiento. Si se conoce una ruta reciente al destino, se regresa un paquete de respuesta de ruta RREP al origen, que le indica cómo llegar al destino. Una ruta se considera reciente si el número de secuencia de destino almacenado en la tabla de enrutamiento es mayor que o igual al número de secuencia de destino del paquete de solicitud de ruta. Si es menor, la ruta almacenada es más antigua que la que el origen tenía para el destino, por lo que se ejecuta el paso 3.

3. Puesto que el receptor no conoce una ruta reciente al destino, incrementa el campo cuenta de saltos y vuelve a difundir el paquete de solicitud de ruta. También extrae los datos del paquete y los almacena como una entrada nueva en su tabla de rutas invertidas. Esta información se utiliza para construir la ruta invertida a fin de que la respuesta pueda regresar posteriormente al origen. También se inicia un temporizador para la nueva entrada de ruta invertida. Si el temporizador expira, la entrada se borra.

Cuando el paquete de solicitud difundido llega al nodo destino, éste construye un paquete de respuesta de ruta. El formato del paquete RREP se muestra en la figura 1.2. Los campos dirección de origen y dirección de destino se invierten respecto a la solicitud entrante, y el número de secuencia de origen se copia de su contador en memoria. El campo cuenta de saltos se inicializa en 0, mientras que el campo tiempo de vida determina el tiempo en milisegundos para el cual los nodos que reciben el mensaje RREP consideran la ruta válida.

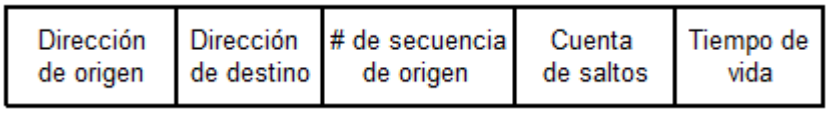


Figura 1.2. Formato del paquete de respuesta de ruta (Fuente: [15]).

El paquete se encamina a través de la ruta inversa, siendo inspeccionado por cada nodo intermedio, y se introduce en la tabla de enrutamiento local como una ruta al destino si se cumple una o más de las siguientes condiciones:

1. No se conoce una ruta al destino (generador del paquete RREP)

2. El número de secuencia de origen en el paquete de respuesta de ruta es mayor que el valor en la tabla de enrutamiento.

3. Los números de secuencia son iguales, pero la nueva ruta es más corta.

Debido a que es posible mover o apagar los nodos, la topología puede cambiar de manera espontánea. El algoritmo debe ser capaz de manejar cualquiera de estas circunstancias. Para ello, cada nodo difunde de manera periódica un mensaje de saludo (*Hello*), que espera que cada uno de sus vecinos responda. Si no se recibe ninguna respuesta, el difusor sabe que el vecino se ha movido del alcance y ya no está conectado a él. De manera similar, si el difusor trata de enviar un paquete a un vecino que no responde, se da cuenta de que el vecino ya no está disponible. Esta información se utiliza para eliminar rutas que ya no funcionan.

Para cada destino posible, cada nodo N mantiene un registro de los vecinos que le han proporcionado un paquete para el destino en cuestión durante los últimos ΔT segundos. Éstos se llaman vecinos activos de N para ese destino. Para gestionar toda esta información, el nodo N mantiene una tabla de enrutamiento codificada por destino que contiene el próximo salto a utilizar, la cuenta de saltos, el número de secuencia de destino más reciente, y la lista de vecinos activos para cada destino.

Cuando cualquiera de los vecinos de N se vuelve inalcanzable, N verifica su tabla de enrutamiento para ver cuáles destinos tienen rutas en las que se incluya al vecino ahora perdido. Para cada una de estas rutas, se les informa a los vecinos activos que su ruta a través de N ahora es inválida, y que se debe eliminar de sus tablas de enrutamiento. Seguidamente, los vecinos activos indican este hecho a sus vecinos activos, y así sucesivamente, de manera recursiva, hasta que las rutas que dependen del nodo perdido se eliminan de todas las tablas de enrutamiento [15].

AODV ha servido de inspiración para el desarrollo de otros protocolos específicos para redes de sensores inalámbricos. Dentro de estos protocolos se pueden mencionar a REL (*Routing by Energy and Link Quality*), que provee un esquema de encaminamiento consciente de la energía, y nst-AODV (*not so tiny-AODV*), una versión simplificada de AODV.

1.2.2 SPIN

Sensor Protocol for Information Via Negotiation (SPIN) es una familia de protocolos centrados en los datos, y desarrollados específicamente para redes de sensores inalámbricos. Es diseñado sobre la base de dos ideas fundamentales. Primeramente, si bien intercambiar la información sensada puede ser una actividad costosa, intercambiar información sobre la información sensada no necesariamente lo es.

En segundo lugar, los nodos deben adaptarse a cambios en su disponibilidad de recursos para extender su vida útil y la de la red. Sus autores afirman que se toman mejores decisiones de encaminamiento si se tiene conocimiento sobre los datos de la aplicación y el estado de los recursos en cada nodo, además de la topología de la red.

Para poder superar las deficiencias del *flooding* clásico (implosión, superposición y ceguera de recursos), SPIN incorpora dos técnicas innovadoras en su momento [7]:

- Negociación: Para evitar la implosión y la superposición, los nodos negocian entre sí la transmisión de los datos, utilizando meta-datos para describir la información disponible.
- Adaptación de recursos: Cada nodo dispone de un administrador de recursos que mantiene registro del consumo de recursos. Cuando la energía es baja, el nodo se abstiene de participar en ciertas actividades, por ejemplo, el envío de información de terceros.

SPIN asume una topología de red plana, donde cualquier nodo sensor puede hacer función de sumidero. Asimismo, cualquier nodo puede ser fuente de datos, es decir, genera información a partir del sensado de un parámetro del medio, y la información debe ser diseminada a través de la red.

Los nodos se comunican mediante tres tipos de mensajes:

1. ADV: aviso o anuncio de nuevos datos. Cuando un nodo dispone de nueva información, lo informa mediante un mensaje ADV que contiene metadatos.
2. REQ: petición de datos. Cuando un nodo recibe un mensaje ADV, verifica si ya ha recibido la información que describe el metadato. En caso negativo, responde con un mensaje REQ pidiendo la nueva información, que será enviada en un mensaje DATA.
3. DATA: mensaje de datos. Contiene la información propiamente dicha, encabezada por el metadato.

Es importante aclarar que en este protocolo no todos los nodos necesitan responder a los mensajes que reciben, por ejemplo, un nodo que recibe un mensaje ADV de un vecino no deberá responder con REQ si ya dispone de la información descrita en el metadato.

A continuación se lista algunos de los protocolos de la familia SPIN:

- SPIN-PP (SPIN *point-to-point*): Es el más básico de los protocolos de la familia SPIN. Se diseña optimizado para una red teórica que utiliza un medio de transmisión punto a punto, asumiendo que es posible que dos nodos A y B se comuniquen exclusivamente uno con el otro, sin interferir la comunicación con otros nodos. SPIN-PP trabaja en tres etapas, cada una de las cuales corresponde a uno de los mensajes descriptos anteriormente.
- SPIN-EC: Agrega lógica de conservación de energía bastante simple a SPIN-PP. Cuando un nodo observa que su energía residual se aproxima a un umbral inferior de carga de batería, reduce su participación en el protocolo. Si el nodo recibe un dato nuevo, sólo iniciará la ronda de negociación si puede completar todas las etapas (anuncio de datos y respuesta de petición). De la misma manera, si recibe un anuncio, sólo enviará una petición si tiene energía suficiente para recibir el dato.
- SPIN-BC: es una versión de SPIN que agrega optimizaciones para cuando se utiliza un canal de transmisión compartido. Cuando un nodo envía un mensaje, éste es recibido por todos los nodos dentro del rango de alcance de la radio, sin importar el destino original de mensaje.

1.2.3 LEACH

Low Energy Adaptive Clustering Hierarchy (LEACH) es un protocolo que propone el uso de *clusters* de nodos permitiendo distribuir el consumo de energía de manera más uniforme entre los sensores de la red. Se caracteriza por hacer rotar al azar el rol de cabecera de *cluster*. Dentro de un *cluster*, la cabecera coordina localmente la transmisión y fusión de datos para comprimir la información luego transmitida directamente a la estación base, reduciendo el ancho de banda requerido [16]. La topología resultante se esquematiza en la figura 1.3.

El modelo de red para el cual se desarrolla LEACH comprende cientos o miles de nodos sensores económicos y eficientes en términos de energía. Los nodos son homogéneos y se encuentran restringidos en cuanto a la duración de la batería. La calidad de la información se mejora con la utilización de un gran número de nodos.

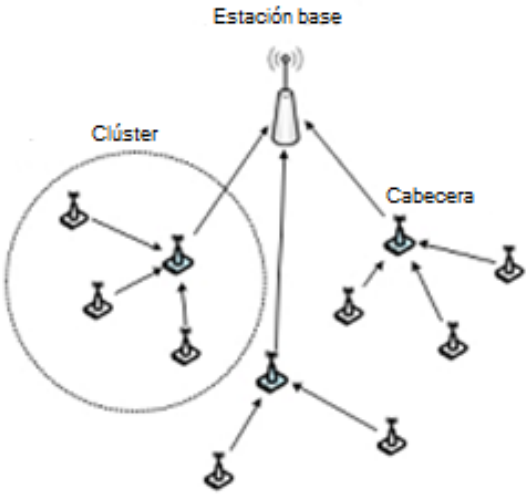


Figura 1.3. Topología de red en LEACH (Fuente: [6]).

LEACH opera en rondas divididas en las siguientes cuatro fases [6]:

1. Anuncio

En la fase de inicialización, cada clúster decide si se vuelve cabecera dependiendo del porcentaje de cabeceras que debe tener la red (definido a priori) y el número de veces que el nodo ya ha tenido el rol. Cada nodo que se elige a sí mismo como cabecera hace *broadcast* del anuncio al resto de los nodos, utilizando CSMA (*Carrier Sense Multiple Access*). Los nodos que no son cabecera mantienen su radio encendida, y elijen su cabecera sobre la base de la fuerza de la señal recibida de los anuncios que escuchan.

2. Armado del *cluster*

Los nodos eligen la cabecera que requiere la menor energía de transmisión, e informan a la cabecera seleccionada que se harán miembro del *cluster* usando el protocolo CSMA.

3. Planificación del canal

Una vez que la cabecera recibe todos los mensajes de los nodos miembros, crea un *schedule TDMA* (*Time Division Multiple Access*) donde asigna un período de tiempo de transmisión a cada nodo, y lo informa por difusión a sus miembros.

4. Transmisión de datos

Cada nodo envía la información durante la franja de tiempo de transmisión que le fue asignada. Mientras el nodo no está transmitiendo, puede apagar su radio. Cuando el nodo cabecera recibe los datos de todos los nodos del clúster, puede combinarlos para comprimir la señal que se enviará a la estación base en una transmisión de alta energía.

Luego de un determinado tiempo de transmisión, comienza una nueva ronda. Para minimizar el costo indirecto asociado al mantenimiento de la jerarquía, la fase de transmisión es mucho más larga que la de armado de clústeres.

LEACH es un algoritmo distribuido y no requiere coordinación de la estación base ni conocimiento global de la topología de red. Es eficiente al distribuir el consumo de energía de manera uniforme a lo largo de la red, lo que incrementa la vida útil de la misma. Dado que las cabeceras pueden comprimir o fusionar los datos recibidos de los nodos locales, contribuyen a reducir el tráfico global.

A pesar de las múltiples propuestas, no existe un protocolo de enrutamiento adecuado para todos los escenarios de aplicación. La continua necesidad de encontrar nuevas soluciones, y de tomar decisiones óptimas en base a diferentes métricas conlleva a la introducción de técnicas de Inteligencia Computacional al enrutamiento en redes de sensores inalámbricos.

1.3 Algoritmos de enrutamiento basados en IC

Diferentes paradigmas de Inteligencia Computacional han encontrado aplicación en el diseño de algoritmos de enrutamiento para ambientes complejos y dinámicos como las redes de sensores. Estos algoritmos exhiben un comportamiento adaptativo y suelen ser robustos ante cambios de topología y fallos en la comunicación. Las técnicas más utilizadas son aquellas basadas en aprendizaje por refuerzos, inteligencia de enjambres, algoritmos

evolutivos y lógica difusa. Se pueden encontrar otras técnicas dentro del campo del aprendizaje de máquina, pero no son adecuados para su aplicación en redes de sensores debido a que sus propiedades no se adecúan a los requerimientos de estos sistemas [9].

1.3.1 Algoritmos basados en redes neuronales

Una red neuronal es el intento de poder realizar una simulación computacional del comportamiento de partes del cerebro humano mediante la réplica en pequeña escala de los patrones que éste desempeña para la formulación de resultados a partir de los sucesos percibidos. Son sistemas paralelos para el procesamiento de la información, inspirados en el modo en el que las redes de neuronas biológicas del cerebro procesan información [17]. Una característica fundamental de las redes neuronales es que tienen altos requerimientos computacionales, por lo que no se han desarrollado muchos algoritmos de enrutamiento basados en este paradigma.

En [18] se presenta un algoritmo de enrutamiento manejado por QoS denominado SIR (*Sensor Intelligence Routing*), donde en cada nodo usa una red neuronal para manejar el tráfico de datos. En el artículo se propone una modificación al algoritmo de Dijkstra para formar el troncal de la red y encontrar los caminos de menor costo desde la estación base hacia cada uno de los nodos sensores.

A través de un mapa auto-organizado (SOM por sus siglas en inglés), que es una red neuronal de aprendizaje no supervisado, se puede determinar el peso de una arista w_{ij} que conecta un par de nodos en base a tres parámetros de QoS: latencia, tasa de errores, y *throughput*. Una vez que un nodo ha recopilado suficientes muestras, es capaz de hacer una estimación de la calidad de servicio de cada enlace.

El rendimiento de SIR es comparado con el de los protocolos *Directed Difussion* y *Energy Aware Routing*. En ambos casos, SIR mostró un rendimiento superior en cuanto a latencia promedio y consumo de energía, sin embargo, la propuesta es costosa si se tiene en cuenta que cada nodo debe hacer un *ping* a sus vecinos para determinar los parámetros del enlace. Además, la implementación de un mapa auto-organizado en nodos sensores conlleva a un gasto computacional significativo que puede impactar en la limitada capacidad de memoria, procesamiento y energía de las motas [9].

1.3.2 Algoritmos basados en lógica difusa

La lógica difusa es una lógica multievaluada que permite que valores intermedios sean definidos dentro de valores umbrales convencionales. En este aspecto, se asemeja al

razonamiento humano, que es capaz de incluir una medida de imprecisión o incertidumbre, que es marcada por el uso de variables lingüísticas tales como *muy, mucho, frecuentemente, poco, muy poco,* etc. Los sistemas difusos permiten el uso de conjuntos difusos para sacar conclusiones y tomar decisiones para optimizar algún parámetro.

Una elección acertada de las cabezas de *cluster* puede reducir el consumo de energía y extender el tiempo de vida de la red. Un algoritmo de lógica difusa basado en la energía, concentración y centralización es propuesto en [19] para la elección de la cabeza de *cluster*. El estudio usa un modelo de red en el cual todos los nodos sensores transmiten a la estación base información sobre su ubicación y energía disponible.

La estación base entonces toma en cuenta, junto con estos parámetros, el número de nodos en la vecindad y la distancia entre los mismos, para determinar cuáles son los nodos adecuados para ejercer la función de cabecera. Para ello, clasifica las variables energía del nodo y concentración del nodo en tres niveles: bajo, medio y alto. De modo similar lo hace con la variable de centralización del nodo en cerca, adecuado y lejano. El resultado difuso que representa la probabilidad de que un nodo sea elegido como cabeza de *cluster* es dividido en siete niveles que sirven como indicador de clasificación.

En el artículo se observa un incremento sustancial en la vida de la red en comparación con una red que usa el protocolo LEACH. Para un escenario de 20 nodos ubicados en un campo de 100 m x 100 m, el número de rondas de datos antes de la caída del primer nodo es en promedio 1.8 veces mayor que en LEACH. Sin embargo, esta alternativa supone un gasto de recursos en recolectar la información necesaria en la estación base antes de determinar las cabezas de *cluster*.

1.3.3 Algoritmos evolutivos (AE)

Los algoritmos evolutivos modelan la evolución natural, que es el proceso de adaptación en vista de mejorar las capacidades de supervivencia a través de procesos como la selección natural, supervivencia del más fuerte, reproducción, mutación, competición y simbiosis. Los AE usan una población de candidatos llamados cromosomas que se reproducen, tomando sus hijos diferentes características. Los cromosomas que mejor se adaptan son seleccionados para formar una nueva generación, los otros son eliminados. El proceso se repite generación tras generación, hasta obtener una solución lo suficientemente aceptable [17].

En [20] se propone un protocolo de enrutamiento multi-salto llamado *GA-Routing* (*Genetic Algorithm Routing*). Está basado en un algoritmo genético y su objetivo es maximizar la longevidad de la red en términos del tiempo transcurrido antes de la caída del primer nodo. El algoritmo propuesto trabaja generando árboles de agregación que abarcan todos los nodos de la red.

Aunque el mejor árbol de agregación es el camino más eficiente hacia la estación base, el uso continuo del mismo puede conllevar al fallo de un grupo de nodos antes que otros. El objetivo entonces es encontrar un árbol de agregación y el número de veces a usar antes de crear otro árbol. Dadas las localizaciones de los nodos sensores y de la estación base, el algoritmo genera una secuencia de caminos para el enrutamiento, y es capaz de reducir el número de transmisiones gracias a una técnica de agregación de datos. De esta manera, se asegura un menor consumo energético en las tareas de comunicación.

Los resultados de la simulación muestran que *GA-Routing* provee un mayor tiempo de vida de la red que un algoritmo simple que siempre selecciona el mejor árbol. Sin embargo, es un protocolo centralizado, y no se consideran los costos de diseminación de los caminos. A esto se debe agregar que las simulaciones se realizan sobre una aplicación en Java, y no sobre un simulador de redes, donde se obtienen resultados más reales.

Otro algoritmo de enrutamiento basado en un algoritmo genético es *Two-TierGA* (*Two-Tier Genetic Algorithm*). Está diseñado para maximizar el tiempo de vida de una red de dos niveles. En este tipo de redes un grupo de sensores actúan como cabezas de *cluster*, y si alguno de estos nodos falla debido a falta de energía en la batería, el *cluster* completo deja de operar. Esto influye en la carga que deben manejar los nodos restantes, que agotan su batería rápidamente [21].

Un cromosoma es representado como una cadena de números de nodos para un esquema de enrutamiento en particular, y la longitud del mismo es igual al número de nodos de relevo. El nodo en el cromosoma que disipa el máximo de energía es marcado como un nodo crítico, y los mensajes dirigidos a él son redirigidos hacia algún otro nodo elegido al azar. Mediante este método se observa en el artículo una extensión de un 200% del tiempo de vida de la red en comparación con otras redes que emplean modelos de mínima energía de transmisión y mínima cantidad de saltos. Sin embargo, las simulaciones se realizan sobre una aplicación en Java, por lo que no se puede considerar la influencia de la subcapa MAC en el consumo de energía de los nodos.

1.3.4 Algoritmos basados en inteligencia de enjambres

La inteligencia de enjambres se originó del estudio del comportamiento colectivo de sociedades de especies biológicas tales como bandadas de pájaros, cardúmenes de peces y colonias de hormigas. En esencia, es la propiedad de un sistema donde los comportamientos colectivos de agentes interactuando localmente con su medio ambiente causan la emergencia de patrones globales.

Un algoritmo de enrutamiento basado en inteligencia de enjambres es *ACO-QoSR* (*Ants Colony Optimization based QoS Routing algorithm*), que utiliza un enfoque del algoritmo de optimización de colonias de hormigas, modificado para resolver problemas de encaminamiento. Toma en cuenta el retardo, las limitantes de energía, y el ancho de banda de la comunicación inalámbrica. ACO-QoSR busca las mejores rutas que satisfagan los requerimientos mínimos de QoS, y el equilibrio entre el cuidado de las restricciones de QoS y la complejidad computacional.

Cuando un nodo origen ha capturado información del medio, ésta debe ser enviada a la estación base. Primero, el nodo revisa su tabla de enrutamiento para encontrar la ruta apropiada. La fase de prueba de una nueva ruta se realiza sólo si no existen rutas disponibles desde el nodo actual hasta la estación base, mientras que el nodo requiere mantener la información guardada en memoria hasta que se pueda transmitir. Después de haber realizado el proceso de descubrimiento de la ruta nueva, el dato que se guardaba en memoria se manda inmediatamente a la estación base [22].

Un algoritmo basado en la meta-heurística de optimización basada en colonia de hormigas para el enrutamiento de datos en redes de sensores inalámbricos configurados en una topología de malla se desarrolla en [23]. Mediante el uso de una regla que comprende la actualización de feromonas, el algoritmo de colonia de hormigas encuentra rutas de bajo consumo de energía, aumentando así la vida útil de la red.

Ambos algoritmos obtienen resultados superiores en cuanto al tiempo de vida de la red en comparación con algoritmos que utilizan como único parámetro la distancia en saltos hacia el sumidero, pero no se realizan comparaciones con protocolos de enrutamiento convencionales.

1.3.5 Algoritmos basados en aprendizaje por refuerzos

Muchas investigaciones se han realizado en el campo de los algoritmos de enrutamiento conscientes de la energía y de los algoritmos de enrutamiento basados en aprendizaje por

refuerzos. A pesar de que varios de ellos no son específicos para redes de sensores inalámbricos, existen múltiples protocolos que pueden ser beneficiosos en este campo [9].

Uno de los principales algoritmos de enrutamiento basados en aprendizaje por refuerzos es *Q-Routing* [24]. Está basado en el esquema *Q-Learning*, una de las ramas del aprendizaje por refuerzos. Es creado con el objetivo de aprender una política de enrutamiento capaz de balancear la tarea de minimizar el número de saltos que toma un paquete con la posibilidad de congestión en las rutas más utilizadas.

Para ello, experimenta diferentes políticas de enrutamiento y guarda estadísticas para determinar qué decisiones son las más adecuadas para minimizar el tiempo total de envío. Mediante este esquema, cada nodo es capaz de decidir a cuál vecino encaminar paquetes para minimizar el tiempo promedio de envío, basado solamente en información local obtenida mediante realimentación.

El tiempo total que toma un paquete para alcanzar su destino está determinado por la demora total de transmisión en enlaces intermediarios y el tiempo total de espera en cola en los nodos del trayecto. La información de enrutamiento es almacenada en cada nodo en una tabla de valores Q como el tiempo de trayecto estimado desde cada nodo vecino hasta el nodo de destino.

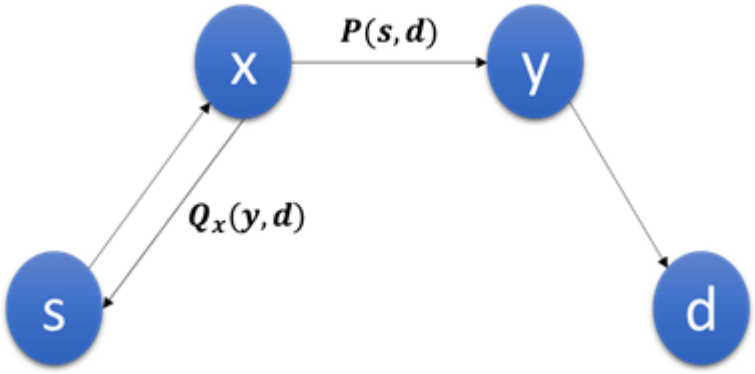


Figura 1.4. Proceso de envío de un paquete en Q-Routing.

Cuando un paquete es enviado, el nodo receptor envía a la fuente un estimado del tiempo restante de trayecto para el paquete en cuestión. Este estimado es interpretado por el nodo fuente como una señal de refuerzo para actualizar el valor Q correspondiente. Después de un período de aprendizaje inicial, los valores Q convergen a los estimados reales, y el estado de red almacenado representará de manera más precisa el estado real de la red, resultando en un mejor rendimiento.

Cada nodo x mantiene una tabla de valores $Q_x(y,d)$, donde $y \in N(x)$ es el conjunto de vecinos de x, y $d \in V$, el conjunto de nodos en la red. Cada valor Q representa el tiempo estimado para que un paquete alcance el destino d desde el nodo y, excluyendo el tiempo de espera en la cola de y. Si los valores Q representan con exactitud el estado actual de la red, la mejor opción es enviar el paquete al vecino con el menor tiempo de entrega estimado para el destino d, esto es lo que se denomina una política local ávida.

La figura 1.4 muestra el esquema de encaminamiento utilizado por *Q-Routing*. Cuando un nodo x recibe un paquete $P(s,d)$ con fuente s y destino d, selecciona el vecino y del vector de valores Q $Q_x(*,d)$, para el cual $Q_x(y,d)$ es el valor mínimo. Posteriormente, el paquete es enviado a y, que inmediatamente responde a x con su mejor estimado τ, que representa el tiempo de trayecto estimado para que un paquete curse desde y hasta el destino d. El cálculo de τ se describe por la ecuación 1.1, donde q_y es el tiempo de espera del paquete en la cola de y.

$$\tau = \min_{z \in vecinos\ de\ y} Q_y(z,d) + q_y \tag{1.1}$$

El nodo x tendrá en cuenta la variable τ durante sus actualizaciones. Cuando un nodo actualiza su tabla local de estimados, también toma en cuenta el tiempo de transmisión del paquete entre los nodos x y y (indicador de la calidad del enlace), representado por s. Posteriormente, se compara el nuevo estimado con el antiguo, y se aplica un factor de aprendizaje η_f a la diferencia (ecuación 1.2). Esto resulta en un valor delta que será usado por x para actualizar el estimado local para el vecino y de acuerdo a la regla de actualización representada por la ecuación 1.3. Este tipo de exploración donde el nodo que envía el paquete recibe un estimado de su vecino se denomina exploración hacia adelante, y es utilizado por varios de los algoritmos de enrutamiento inspirados en *Q-Routing*.

$$\Delta Q_x(y,d) = \eta_f\big(s + \tau - Q_x(y,d)\big) \tag{1.2}$$

$$Q_x(y,d) = Q_x(y,d) + \Delta Q_x(y,d) \tag{1.3}$$

Q-Routing ha servido de inspiración para el desarrollo de otros protocolos de enrutamiento basados en aprendizaje por refuerzo. Tales son los casos de *DRQ-Routing*, *CQ-Routing*, *CDRQ-Routing* y FROMS.

1.3.6 FROMS

En [25] se presenta un protocolo de enrutamiento *multicast* denominado FROMS (*Feedback ROuting to Multiple Sinks*), que explota las potencialidades del aprendizaje por refuerzos. Este protocolo es adecuado para cualquier aplicación de envío periódico de datos hacia uno o varios sumideros en un ambiente multi-salto. Sus principales ventajas son:

- Capacidad de encontrar rutas *multicast* óptimas.
- Incorpora diferentes métricas de costo, lo que lo hace adaptable a diferentes metas de optimización.
- Rápida recuperación ante fallas.

La principal meta de FROMS, según sus creadores, es proveer al desarrollador de una WSN de una única solución de enrutamiento, capaz de configurarse para diferentes escenarios de aplicación. Este es el protocolo de enrutamiento inteligente elegido para su estudio en el presente trabajo, ya que provee un esquema de encaminamiento consciente de la energía, y se aborda con más detalle en el capítulo 2.

Cada paradigma de Inteligencia Computacional tiene criterios de convergencia y requerimientos computacionales particulares. Además, no todos pueden ofrecer soluciones de forma distribuida, que es lo ideal en el enrutamiento para redes de sensores inalámbricos. Otro factor que los diferencia es el tráfico generado para el descubrimiento y mantenimiento de rutas. A continuación se ofrece una breve comparación de los paradigmas de Inteligencia Computacional de acuerdo a los requerimientos para su uso en una WSN.

1.3.7 Comparación entre los paradigmas de Inteligencia Computacional

Las redes neuronales y algoritmos evolutivos tienen altas demandas de procesamiento y son generalmente soluciones centralizadas. En el caso de las redes neuronales, el aprendizaje se puede hacer en línea en cada uno de los nodos, pero es un proceso lento y con altos requerimientos de memoria. La lógica difusa es bastante adecuada para implementar algoritmos de optimización de rutas, sin embargo, no genera soluciones óptimas y nuevas reglas difusas deben aprenderse ante cambios de topología [9].

La inteligencia de enjambres es un paradigma muy popular para computar esquemas de enrutamiento para MANET (*Mobile Ad-hoc Network*). Sin embargo, en las WSN requiere de un alto *overhead* de comunicación para que las “hormigas artificiales” transiten separadamente para el manejo de rutas. Además, generalmente necesita que las

"hormigas" vuelvan a la fuente de datos, lo que a largo plazo se traduce en un gasto significativo de energía [9].

El aprendizaje por refuerzos es la mejor opción para lidiar con problemas distribuidos y dinámicos como el enrutamiento y la agrupación de nodos en las WSN. Tiene exactamente las propiedades necesitadas y ha sido aplicado a ambos problemas. El aprendizaje por refuerzos produce decisiones de enrutamiento óptimas, es robusto y flexible ante fallos de nodos y enlaces, además de ser un algoritmo totalmente distribuido. Sus requerimientos de comunicación son muy bajos y asegura el envío de datos incluso ante cambios en la topología. *Q-Learning* es el método más popular de aprendizaje por refuerzos aplicado a redes de sensores inalámbricos por su flexibilidad, y porque presenta las menores demandas de comunicación y procesamiento [9].

Tabla 1.1. Requerimientos de los diferentes paradigmas de IC (Fuente: [9]).

Paradigma	Requerimientos computacionales	Requerimientos de memoria	Flexibilidad
Redes neuronales	Medio	Medio	Baja
Lógica difusa	Medio	Medio	Alta
Algoritmos evolutivos	Medio	Alto	Baja
Inteligencia de enjambres	Bajo	Medio	Alta
Aprendizaje por refuerzos	Bajo	Medio	Alta

1.4 Simuladores

Los simuladores ayudan a los desarrolladores a llevar a cabo experimentos con nuevos algoritmos, protocolos y aplicaciones antes de ser implantados en bancos de pruebas con nodos sensores reales. Algunos de los simuladores son adaptaciones de los simuladores de redes de propósito general con la inclusión de módulos para la simulación de redes de sensores. También existen simuladores para plataformas específicas.

1.4.1 Simuladores de propósito general

Estos simuladores se caracterizan por haber sido diseñados para la simulación de redes tanto cableadas como inalámbricas, y para los que se han creado módulos que permiten llevar a cabo la simulación de redes de sensores. Entre los más populares podemos encontrar:

NS-2

Network Simulator 2 [26] es un simulador de eventos discretos desarrollado en C++. Es uno de los simuladores de redes no específicos más populares y soporta una gran cantidad de protocolos para diversos tipos de redes. Este simulador se basa en el paradigma de la reusabilidad y proporciona una buena colección de protocolos de enrutamiento. Incluye un módulo de aplicación llamado *Network Animator* (nam), que proporciona los resultados de forma visual. Sin embargo, los últimos trabajos investigativos optan por el uso de OMNeT++, ya que posee una interfaz gráfica que facilita mucho el diseño.

OMNeT++

Objective Modular Network Testbed [27], al igual que ns-2, es un simulador de eventos discretos implementado en C++. Proporciona una potente GUI (*Graphical User Interface*) para llevar a cabo el seguimiento y el depurado de las aplicaciones. Está basado en una jerarquía anidada de módulos que proporciona modelos de hardware, incluyendo CPU, batería y radio. En esencia, es un software que provee la infraestructura necesaria para implementar simulaciones de sistemas que pueden ser modelados por eventos discretos.

Existen una serie de *frameworks* y simuladores específicos para llevar a cabo la simulación de redes de sensores, entre los que se encuentran MiXiM, *MobilityFramework* y Castalia. Este último tiene el conjunto más completo de módulos para la simulación de redes de sensores inalámbricos, y cuenta con una amplia comunidad de seguidores que comparte en línea ideas y código fuente.

J-Sim

Java Simulator [28] es un simulador basado en componentes, y desarrollado en Java. Proporciona simulación de procesos en tiempo real. Su mayor ventaja es la cantidad de protocolos soportados, incluyendo un marco de simulación específico para redes de sensores que contiene un modelo muy detallado de estas redes. Los modelos de J-Sim son reutilizables e intercambiables, ofreciendo máxima flexibilidad. Además, proporciona una GUI para mostrar los resultados de simulación y una interfaz que facilita la creación de scripts de simulación. Su principal desventaja es que los tiempos de simulación son mayores en comparación con simuladores basados en C++ [12].

1.4.2 Simuladores para plataformas específicas

El creciente interés por las redes de sensores, y la necesidad de realizar pruebas preliminares, previas a la implementación en bancos de pruebas reales, han llevado al

desarrollo de simuladores para plataformas específicas. A continuación se comentan algunos de los más comentados en la literatura.

TOSSIM

Es un simulador de eventos discretos que está incluido en TinyOS. Simula la ejecución de código nesC en nodos Mica, permitiendo la emulación de nodos reales mediante el mapeado de las interrupciones hardware en eventos discretos. Los componentes de emulación de hardware son compilados junto con los componentes reales de TinyOS, utilizando el compilador de nesC. De esta forma, se obtiene un programa ejecutable real de TinyOS que es ejecutado sobre una capa física simulada [12].

Atemu

Atemu es un simulador y emulador específico para nodos Mica. Durante el funcionamiento del simulador, el modelo de radio es simulado mientras que la operación de los nodos es emulada instrucción por instrucción, permitiendo llevar a cabo pruebas de aplicaciones, protocolos, algoritmos y sistemas operativos, además de la capacidad para simular redes heterogéneas compuestas por diferentes modelos de nodo [12].

La elección del simulador está relacionada con los requerimientos de la investigación que se desea desarrollar. La precisión de los resultados varía de acuerdo al simulador elegido, ya que implementan diferentes modelos de canal de radio, modulación, consumo de energía, etc.

En este capítulo se abordaron las principales características de los protocolos de enrutamiento para redes de sensores inalámbricos. Se explicó el concepto de consciencia de la energía y la importancia de ésta para aumentar el tiempo de vida de una red de sensores inalámbricos.

Se abordaron algunos de los algoritmos de enrutamiento inteligentes para redes de sensores inalámbricos, y se pudo observar que tienen la capacidad de incorporar varias métricas y generar soluciones óptimas. Sin embargo, por lo general tienen altos requerimientos de cómputo o de comunicación.

Del estudio teórico de los algoritmos de enrutamiento basados en Inteligencia Computacional, se concluye que los basados en aprendizaje por refuerzos son los más adecuados para obtener resultados óptimos con bajos requerimientos de hardware y bajo *overhead* de comunicación.

CAPÍTULO 2. IMPLEMENTACIÓN DEL PROTOCOLO INTELIGENTE FROMS

En este capítulo se explica el funcionamiento del protocolo FROMS, el algoritmo de enrutamiento inteligente seleccionado como objeto de estudio. Además, se describen las características de REL, un protocolo de enrutamiento convencional consciente de la energía, que será objeto de comparación con FROMS. Finalmente, se implementa el protocolo FROMS en Castalia, y se realizan modificaciones al simulador para adaptarlo a las necesidades de las simulaciones.

2.1 FROMS

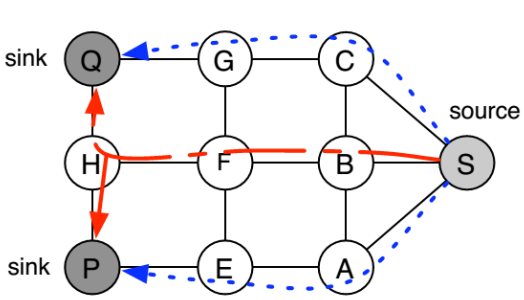
Como se expuso en el capítulo anterior, el objetivo de FROMS es encontrar un camino óptimo para los datos desde su fuente hacia uno o varios sumideros. La optimización se busca sobre la base de una métrica, que puede ser mínima cantidad de saltos o una métrica combinada de saltos y energía residual de las baterías.

Para explicar el funcionamiento de FROMS se toma como referencia la red de la figura 2.1, donde existe una fuente de datos S y dos sumideros: P y Q. Un posible camino desde la fuente hacia los sumideros está formado por la unión de los caminos individuales desde la fuente hacia cada uno de los sumideros (representados por líneas puntuadas), sin embargo, un camino más corto puede existir. Este camino más corto tiene la forma de un árbol, como aquel formado por los nodos B, F y H.

El reto es identificar globalmente este camino sin conocimiento de toda la topología, y usando solamente intercambio de información local. Entonces, la tarea se traduce en actualizar la información local relacionada con los próximos saltos para alcanzar los sumideros desde cada nodo, de manera tal que el costo del árbol resultante sea óptimo.

Durante una fase inicial de anuncios del sumidero, todos los nodos reúnen información inicial de enrutamiento y registran los sumideros conocidos de la red. En el ejemplo de la figura 2.1, el nodo S reúne información de cantidad de saltos para alcanzar cada sumidero individualmente, tal como se muestra en la tabla de enrutamiento. Cuando un paquete de

datos llega al nodo para ser encaminado, éste selecciona uno o más nodos como próximo salto hacia el sumidero.



routing table : node S

Neighbor A	sink P	3 hops
	sink Q	5 hops
Neighbor B	sink P	4 hops
	sink Q	4 hops
Neighbor C	sink P	5 hops
	sink Q	3 hops

Figura 2.1. Topología de ejemplo con dos sumideros y una fuente (Fuente: [25]).

Sin embargo, en lugar de simplemente seleccionar el más indicado en ese momento (en el ejemplo: nodo C para el sumidero Q y nodo A para el sumidero P), también explora rutas sub-óptimas en asunción de que algunas de ellas puedan tener menores costos que los asociados en su tabla de enrutamiento. La razón detrás de este planteamiento está en que nodos vecinos podrían compartir próximos saltos. Por ejemplo, el nodo fuente S estima que desde el nodo A necesita 7 saltos para alcanzar ambos sumideros: 3 saltos para alcanzar P, 5 saltos para el sumidero Q, y el primer salto, que es compartido, por lo que se descuenta.

Por otra parte, el nodo S no conoce si A será capaz de compartir el próximo salto o necesitará dividir el camino a través de dos vecinos diferentes. En el ejemplo, el nodo A es capaz de compartir el próximo salto, y calcula que puede alcanzar los sumideros a través del nodo E (ver la tabla de enrutamiento de la figura 2.1) en (2 + 4) – 1 = 5 saltos. Por tanto, el nodo S será capaz de alcanzar ambos sumideros en un salto hasta el nodo A, más 5 saltos desde A hasta todos los sumideros, para un total de 6 saltos, que es 1 salto menos que la información inicial en el nodo fuente. De ahí que A necesita informar a S sobre su propio estimado del costo de enrutamiento hacia ambos sumideros.

Para ello, asumiendo que todos sus vecinos pueden escuchar sus transmisiones en el canal inalámbrico, el nodo A hace uso de la difusión cuando va a enviar el paquete de datos hacia el próximo salto, y agrega a cuestas su propio estimado de costo. Similarmente, el nodo E añade a cuestas sobre los datos su estimado de costo e informa al nodo A, y así sucesivamente.

2.1.1 Enrutamiento multicast con Q-Learning

Cada nodo sensor es un agente independiente, y las acciones son opciones de enrutamiento usando diferentes vecinos como próximo salto hacia un subconjunto de sumideros $D_p \subseteq D$, listado en el paquete de datos.

En el modelo de FROMS, una acción es una decisión de enrutamiento posible para el paquete de datos. Sin embargo, la información de enrutamiento puede incluir varios nodos vecinos como próximos saltos. Consecuentemente, se necesita realizar modificaciones en el algoritmo *Q-Learning* original, y definir una posible acción a como un conjunto de sub-acciones $\{a_1 \dots a_k\}$. Cada sub-acción $a_i = (n_i, D_i)$ incluye un único vecino n_i y un conjunto de destinos $D_i \subseteq D_p$, indicando que el vecino n_i es el próximo salto deseado para encaminar paquetes hacia los destinos D_i. Una acción completa es un conjunto de sub-acciones tales que $\{D_1 \dots D_k\}$ particiona D_p (esto es, cada sumidero $d \in D_p$ es cubierto por exactamente una sub-acción a_i).

Continuando con el ejemplo de la figura 2.1, considere un paquete destinado a $D_p = \{P, Q\}$. Una posible acción completa de la fuente S es la sencilla sub-acción $(B, \{P, Q\})$, indicando al vecino B como próximo salto para todos los destinos. Alternativamente, el nodo S podría elegir dos sub-acciones: $(A, \{P\})$ y $(C, \{Q\})$, indicando que dos vecinos diferentes deben tomar la responsabilidad de encaminar el paquete a diferentes subconjuntos de vecinos. La distinción entre acciones completas y sub-acciones es importante ya que solamente las sub-acciones reciben recompensa.

Debido a que FROMS está basado en *Q-Routing*, tiene la necesidad de explorar sus vecinos para poder encontrar nuevas rutas con mejores estimados. El algoritmo se apoya en la técnica ϵ-*greedy*, donde cada nodo realiza una acción al azar con una probabilidad ϵ, y selecciona la mejor ruta disponible con probabilidad 1 - ϵ. Cuando un nodo selecciona la mejor acción disponible, se dice que explota la información almacenada, mientras que acciones aleatorias resultan en exploración.

2.1.2 Valores Q

En FROMS el procedimiento de inicialización de los valores Q es diferente al de *Q-Learning*, que inicialmente inicializa los valores Q aleatoriamente. En el caso de FROMS, inicialmente representan el costo real de las rutas, por ejemplo, si la función de costo es el número de saltos, el valor Q de una ruta es también el número de saltos. Para inicializar estos valores, se usa un procedimiento que calcula un estimado del costo basado en la información

individual sobre el vecino esperado y los sumideros. Esta inicialización no aleatoria agiliza significativamente el proceso de aprendizaje y evita oscilaciones de los valores Q.

El primer paso es calcular el valor de las sub-acciones, y luego el de una acción completa. El valor inicial para una sub-acción $a_i = (n_i, D_i)$ está determinado por la ecuación 2.1, donde $saltos_d^{n_i}$ es el número de saltos para alcanzar el destino $d \in D_i$ usando el vecino n_i, y $|D_i|$ es el número de sumideros en D_i.

La primera parte de la ecuación calcula el número total de saltos para alcanzar individualmente los sumideros, y la segunda parte sustrae cierta cantidad de saltos al total, basada en la asunción de que la comunicación por difusión se realiza tanto para transmitir hacia n_i como por n_i para alcanzar al próximo salto. El valor Q de una acción completa a con sub-acciones $\{a_1, \dots, a_k\}$ se calcula mediante la ecuación 2.2, donde k es el número de sub-acciones. Este valor Q es el número de saltos para hacer llegar por difusión un paquete desde el agente hacia todos los sumideros.

$$Q(a_i) = \left(\sum_{d \in D_i} saltos_d^{n_i} \right) - 2(|D_i| - 1) \tag{2.1}$$

$$Q(a) = \left(\sum_{a_i \in a,\ i=1\dots k} Q(a_i) \right) - (k - 1) \tag{2.2}$$

2.1.3 Actualización de valores Q

Para aprender los valores reales de las acciones, el agente debe recibir las recompensas del ambiente. La regla de actualización de valores Q es similar a la usada por el algoritmo *Q-Routing* (ecuación 2.3), donde $R(a_i)$ es el valor de la recompensa.

$$Q_{nuevo}(a_i) = Q_{viejo}(a_i) + \gamma \left(R(a_i) - Q_{viejo}(a_i) \right) \tag{2.3}$$

La recompensa es el valor numérico que permite al nodo en el camino de los datos informar al nodo anterior de su costo actual para la acción solicitada. Por lo tanto, cuando un nodo calcula la recompensa, selecciona su mejor valor Q en la tabla de enrutamiento para el conjunto de destinos, y añade el costo de la acción misma. Este procedimiento se describe por la ecuación 2.4, donde c_{a_i} es el costo de la acción y tiene valor 1. Esta propagación de

valores Q en el camino contrario a los datos eventualmente permite que todos los nodos aprendan los costos reales de las acciones.

$$R(a_i) = c_{a_i} + \min_a Q(a) \tag{2.4}$$

2.1.4 Enrutamiento consciente de la energía con FROMS

Toda técnica de encaminamiento propuesta a trabajar en una WSN debe ser consciente en cuanto a los gastos energéticos, esto es, disminuir los gastos de comunicación y tomar medidas que favorezcan el tiempo de vida de la red. Con la métrica de costo combinada de saltos y energía residual de las baterías, se pretende balancear los gastos energéticos en una red de sensores inalámbricos.

En este caso, los valores Q son función de ambas métricas, como indica la ecuación 2.5, donde E_{saltos} es el estimado de costo en saltos de la ruta y $E_{batería}$ es el costo estimado de batería de la ruta, que se define como la mínima energía residual de todos los nodos a través de la misma (ecuación 2.6).

$$Q_{comb}(ruta) = f(E_{saltos}, E_{batería}) \tag{2.5}$$

$$E_{batería}(ruta) = \min_{n_i \in ruta} batería \tag{2.6}$$

$$f(E_{saltos}, E_{batería}) = hcm(E_{batería}) * E_{saltos} \tag{2.7}$$

La función f que combina ambos estimados en un solo valor Q se define por la ecuación 2.7, donde hcm (*hop count multiplier*) es el multiplicador del número de saltos, y es generado por una función que pesa el estimado de número de saltos en base a la energía residual de la batería del vecino. Mientras menor sea este valor, mayor será el multiplicador. Esto se hace con el objetivo de desalentar futuros usos de rutas que involucran nodos con un bajo nivel de energía residual, y extender así el tiempo de vida de la red.

La figura 2.2 muestra cuatro funciones de costo diferentes (también conocidas como funciones de ponderación). Si el nivel de batería es irrelevante, entonces $hcm(E_{batería})$ es constante y $f(E_{saltos}, E_{batería})$ se reduce a una función basada en saltos solamente, donde $hcm = 1$.

Por el otro lado, si se desea encontrar un balance entre la distancia en saltos y el consumo de energía, se utiliza una función de costo dinámica. Se consideran tres funciones de este tipo: dos lineales (ecuaciones 2.8 y 2.9) y una exponencial (ecuación 2.10). Cada una de

ellas genera un rango de multiplicadores diferentes, por lo que el criterio de elección depende de los requerimientos de la aplicación. Los valores de $energía\_disponible$ se encuentran en el rango de 0 y 1, con $energía\_máxima = 1$.

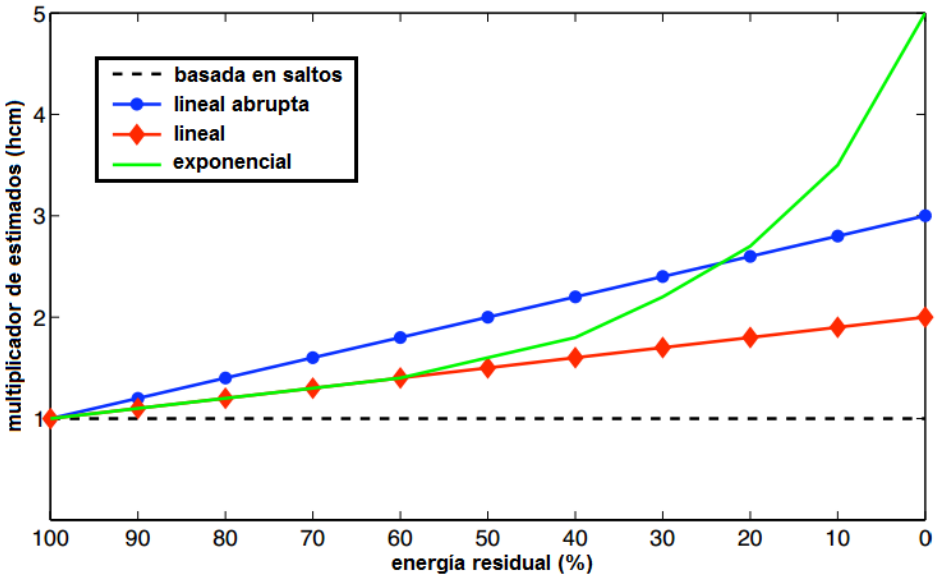


Figura 2.2. Multiplicadores de estimados para diferentes funciones de costo.

$$w_1(energía\_disponible) = (2 * energía\_máxima) - energía\_disponible \quad (2.8)$$

$$w_2(energía\_disponible) = (3 * energía\_máxima) - energía\_disponible \quad (2.9)$$

$$w_3(energía_{disponible}) = 5^{(energía\_máxima\ -\ energía\_disponible)} \quad (2.10)$$

2.1.5 Manejo de fallas

Cada nodo mantiene una estructura de datos llamada SinkControl, cuyo objetivo es detectar enlaces caídos. Dicha estructura almacena información sobre cada sumidero conocido por el nodo. La figura 2.3 representa un ejemplo de SinkControl para el nodo E de la red de ejemplo de la figura 2.1.

En la realimentación se incluye una estampilla de tiempo sobre la última vez que el nodo ha recibido información sobre el sumidero. Si esta estampilla es demasiado vieja (excede un determinado umbral), se invalidan las rutas hacia el sumidero. Esto sucede cuando el sumidero ha caído, o cuando fallan los enlaces en un punto determinado de la red, y no se pueden encaminar los datos hacia su destino.

Para adaptarse a fallas en los enlaces, cada nodo almacena la última vez que escuchó a cualquier vecino transmitir un paquete. Adicionalmente, almacena la última vez que

encaminó un paquete hacia ese vecino. En caso de que la diferencia entre ambas estampillas exceda un umbral determinado, el vecino es eliminado de la tabla. Si esto sucede por error, la próxima vez que el nodo escuche a este vecino recuperará la ruta. A diferencia de algunos protocolos que usan manejo de vecinos, FROMS no se auxilia de balizas o difusiones periódicas. En su lugar, se auxilia de la escucha de los paquetes de datos transmitidos por los vecinos para chequear la disponibilidad de los mismos.

SinkControl : node 7

sink	last timestamp	direct neighbor	direct timestamp
sink 1	-2 sec	true	-2 sec
sink 2	-14 sec	false	-

Figura 2.3. Estructura SinkControl para el nodo E (Fuente: [25]).

Para el estudio de FROMS es necesario hacer una comparación con un protocolo de enrutamiento convencional. Se elige al protocolo REL, ya que también incorpora realimentación sobre la energía residual de los nodos y utiliza reglas que permiten balancear el consumo de energía.

2.2 REL (*Routing by Energy and Link quality*)

REL [29] es un protocolo de enrutamiento para redes planas cuyo objetivo es incrementar la confiabilidad y eficiencia en el consumo de energía en una red de sensores inalámbricos. REL selecciona rutas en base a un mecanismo de estimación de la calidad del enlace de extremo a extremo, energía residual y cantidad de saltos. Además, propone un mecanismo manejado por eventos para la distribución de la carga y evitar el agotamiento prematuro de la energía de los nodos.

2.2.1 Estimación de la calidad del enlace

Para analizar un enlace, REL se apoya del LQI (*Link Quality Indicator*), que es una métrica proveída por la capa física del estándar IEEE 802.15.4. LQI varía desde 0 hasta 255, y es calculada en base a la RSSI (*Received Signal Strength Indication*), SNR (*Signal-Noise-Ratio*) o una combinación de ambas métricas. Sin embargo, el valor de LQI solamente refleja la calidad del enlace para los nodos vecinos, no de extremo a extremo. Para determinar la calidad del enlace hacia el destino, REL propone el uso de *WeakLinks*, que no es más que un contador de saltos para los peores enlaces a través de un camino determinado. En términos específicos, si el LQI para un determinado enlace es menor que

LQI_{th}(*LQI threshold*), el enlace se considera débil y el contador *WeakLinks* es incrementado.

WeakLinks es incorporado en los mensajes RREQ (*Route Request*) y RREP (*Route Reply*), y posteriormente actualizado en cada salto durante el proceso de descubrimiento de rutas. Cada vez que un nodo recibe un mensaje RREQ o RREP, actualiza su valor de LQI. Seguidamente, debe determinar si dicho valor de LQI es menor que LQI_{th}, y actualizar *WeakLinks* si es necesario.

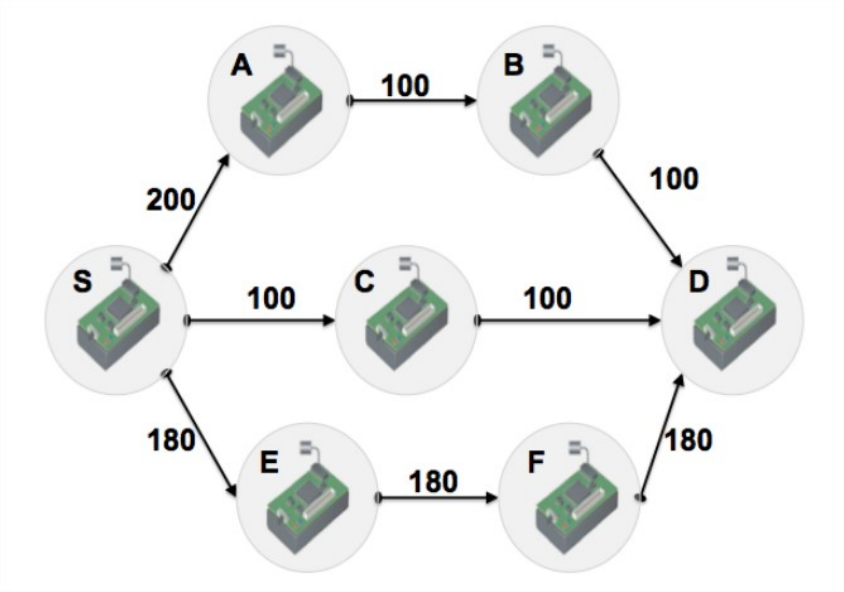


Figura 2.4. Estimación de la calidad del enlace extremo a extremo (Fuente: [29]).

Asuma que LQI_{th}= 170 en la red de ejemplo de la figura 2.4, donde la fuente es el nodo S y el destino el nodo D. Un proceso de selección de rutas que elige el camino de menor distancia en saltos seleccionará al nodo C como próximo salto. Sin embargo, este nodo posee enlaces de baja calidad, y es más susceptible a causar pérdidas de paquetes que los otros enlaces.

Por el otro lado, un mecanismo de selección de rutas que use solamente la calidad del enlace como métrica, seleccionará al nodo A como el próximo salto, aunque existen caminos poco confiables en el camino restante. Finalmente, cuando se utiliza *WeakLinks* como métrica, puede determinarse que los nodos A y B poseen 2 enlaces débiles cada uno, y que el nodo E no posee enlaces débiles. Por lo tanto, cuando los enlaces son analizados y seleccionados desde la perspectiva de la calidad extremo a extremo, el camino más

confiable es a través del nodo E. Esta ruta es capaz de proporcionar mayor confiabilidad para la entrega de datos.

2.2.2 Selección de rutas y balanceo de cargas

REL explota un esquema reactivo para encontrar rutas bajo demanda con el objetivo de reducir los gastos de señalización. El proceso de descubrimiento de rutas incluye la difusión de mensajes RREQ y RREP. Estos mensajes buscan rutas disponibles y asisten al proceso de selección de rutas mediante la recolección de información sobre la energía residual y la calidad del enlace. Cada mensaje RREP representa una ruta disponible hacia el destino, y de acuerdo a la configuración de REL, es posible almacenar n rutas posibles hacia cada nodo de destino. REL integra/administra los valores de tres métricas claves para encontrar las mejores rutas:

- Calidad de los enlaces inalámbricos en base a *WeakLinks*.
- Energía residual de la batería.
- Distancia en saltos, para evitar caminos largos e ineficientes.

El proceso de selección de rutas depende de dos umbrales para establecer comparaciones entre las posibles rutas. El primero de ellos es el umbral de conteo de saltos $HCdiff_{\max\_allow}$ (*Hop Count Maximum Difference*), que determina cuál es la máxima diferencia en saltos para determinada ruta. El segundo umbral es E_{th} (*Energy Threshold*), un umbral de energía residual que es utilizado en dos fases: en el proceso de selección de rutas y en el mecanismo de balanceo de cargas.

En el caso del mecanismo de balanceo de cargas, E_{th} responde al seguimiento de los niveles de energía observados en cada nodo individualmente. Cuando la red inicia su ejecución (fase de *bootstrap*), cada nodo debe almacenar su porcentaje de energía residual y, cada t unidades de tiempo, debe comparar el nivel de energía actual E_t con el que había sido almacenado previamente (E_{t-1}). Si la diferencia entre ambos valores es mayor que E_{th}, se activa un evento en respuesta.

La diferencia entre valores de energía responde al nombre de Ind_{RADV} (*index of Route Advisor*). Si $Ind_{RADV} > E_{th}$, se envía un mensaje de notificación RADV (*Route Advisor*) hacia los nodos vecinos. Este mensaje contiene información sobre el nuevo valor de energía residual del nodo en cuestión, y advierte a los nodos vecinos que deben evaluar nuevamente el uso de dicho nodo en sus rutas.

La figura 2.5 muestra el algoritmo de selección de rutas de REL, donde existen tres reglas básicas. Estas reglas filtran las rutas de acuerdo a las sentencias condicionales de las líneas 3, 9 y 15, que evalúan la ruta candidata R_b y la categorizan de acuerdo a su nivel de energía, comparándola con la ruta activa R_a.

```
1:  Considere Ra = ruta activa
2:  Considere Rb = ruta alternativa
3:  if Ra.energía = Rb.energía then
4:      if Ra.saltos > Rb.saltos + HCdiff_max_allow then
5:          if Ra.weakLinks ≥ Rb.weakLinks then
6:              SeleccionarRuta(Rb)
7:          end if
8:      end if
9:  else if Ra.energía < Rb.energía then
10:     if Ra.saltos + HCdiff_max_allow ≥ Rb.saltos then
11:         if Ra.weakLinks ≥ Rb.weakLinks then
12:             SeleccionarRuta(Rb)
13:         end if
14:     end if
15: else if Ra.energía > Rb.energía and Ra.energía ≤ Rb.energía + Eth then
16:     if Ra.weakLinks ≥ Rb.weakLinks and Ra.saltos > Rb.saltos + HCdiff_max_allow then
17:         SeleccionarRuta(Rb)
18:     end if
19: end if
```

Figura 2.5. Algoritmo de selección de rutas de REL (Fuente: [29]).

La línea 15 muestra el uso de E_{th} como un parámetro de tolerancia para la diferencia que es aceptable si R_b tiene menor energía. Después de analizar el nivel de energía, el algoritmo calcula el número de saltos y evalúa la calidad de los enlaces. Este paso ocurre en las líneas 5, 10 y 16, donde el umbral utilizado es $HCdiff_{\max\_allow}$. Si ambos casos son ciertos, entonces se conmuta hacia la ruta R_b (líneas 6 y 12).

La regla descrita en la línea 15 es un caso especial en el proceso de selección, ya que analiza los casos en los que R_b tiene mejor energía que la ruta activa. En este caso, R_b debe reemplazar a R_a si la diferencia con respecto al umbral E_{th} y R_b es considerablemente pequeña (línea 16).

El análisis del desempeño de REL en comparación con AODV fue realizado en el simulador Castalia. Los resultados de las simulaciones muestran que REL incrementa el tiempo de vida de la red (en función del tiempo de caída del primer nodo) en hasta un 26.6% en comparación con AODV [29].

Para la comparación de FROMS contra REL en cuanto al tiempo de vida de la red, se implementan ambos en un simulador. Como se comentó en el capítulo anterior, la elección del simulador está determinada por las necesidades específicas de la investigación. Consecuentemente, para la evaluación de la capacidad de un protocolo de enrutamiento de incrementar la vida de la red, es necesario que el simulador considere los gastos energéticos de comunicación.

Para la implementación de los protocolos se utiliza Castalia, que posee un modelo de consumo de energía realístico basado en el estado y potencia de transmisión de transceptores reales. Además, está disponible el código fuente escrito por los desarrolladores de REL.

2.3 Castalia

Castalia es un simulador para redes de sensores inalámbricos, redes de área personal, y en general, redes de dispositivos de baja potencia. Está basado en la plataforma de OMNeT++ y es compatible con varias versiones de Linux. Castalia puede ser utilizado por investigadores y desarrolladores que desean probar sus algoritmos distribuidos y/o protocolos en canales inalámbricos y modelos de radio realísticos. También puede ser usado para evaluar plataformas en aplicaciones específicas, ya que es altamente paramétrico y puede simular un amplio rango de plataformas [30]. Sus principales características son:

- Modelo avanzado del canal basado en datos obtenidos empíricamente.
- El modelo define un mapa de las pérdidas del trayecto, en vez de conexiones simples entre los nodos.
- Soporta movilidad de los nodos.

- Modelo avanzado de radio basado en sistemas reales para comunicaciones de baja potencia.
- Múltiples niveles de potencia de transmisión.
- Estados con diferentes consumos de potencia y retardos en la conmutación entre estados.
- Ruido del dispositivo sensor, rango de error y consumo de potencia.
- Protocolos de enrutamiento y MAC disponibles.
- Diseñado para la adaptación y expansión.

Castalia fue diseñado para que los usuarios puedan implementar sus algoritmos y protocolos haciendo uso de las librerías y módulos que provee el simulador. No es una plataforma específica para ningún tipo de sensor. Su objetivo es proveer una arquitectura genérica, confiable y realística para la evaluación de algoritmos antes de proceder a la implementación en bancos de prueba reales [30].

2.3.1 Estructura básica

Castalia está basado en OMNeT++, por tanto, tiene una estructura jerárquica basada en módulos. Cada nodo es un módulo compuesto, cuya estructura se muestra en la figura 2.6. Los nodos no se conectan directamente entre ellos, sino a través del módulo del canal inalámbrico. Por el otro extremo se enlazan al proceso físico, del cual obtienen información sobre alguna magnitud física.

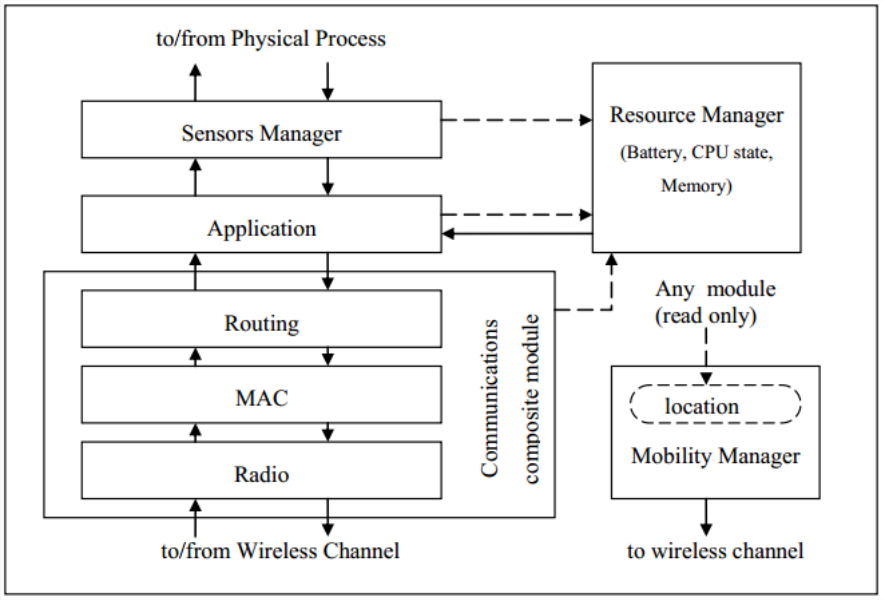


Figura 2.6. Módulo compuesto para los nodos en Castalia (Fuente: [30]).

El software ofrece, en forma de clases abstractas, las herramientas necesarias para construir nuevos protocolos o aplicaciones. Todos los módulos existentes son configurables a través de varios parámetros.

2.3.2 *Scripts* de simulación y recopilación de resultados

OMNeT++ permite seleccionar solamente una de las configuraciones especificadas en el archivo de configuración de la simulación, generalmente nombrado omnetpp.ini. Castalia ofrece la ventaja de combinar varias configuraciones a través de un *script* (Castalia) programado en Python. De esta manera se pueden variar los parámetros de los módulos dinámicamente, por ejemplo la potencia de transmisión de los transceptores, o la tasa de datos en cada repetición de la simulación. El *script* llama al ejecutable (CastaliaBin.exe) con los parámetros necesarios y se ejecuta el programa de simulación. Como resultado se obtienen dos nuevos ficheros:

- Castalia-Trace.txt: contiene las trazas de los eventos ocurridos en cada módulo.
- YYMMDD-HHMMSS.txt: es el archivo de salida estándar de Castalia y contiene los resultados recopilados por cada módulo. El nombre se corresponde con la fecha y hora actual del sistema en que fue creado.

Castalia no provee interfaz gráfica, todas las operaciones deben realizarse desde la línea de comandos de Linux. El archivo de salida puede ser leído directamente en un procesador de texto, pero lo más adecuado es procesarlo a través del *script* CastaliaResults. El *script* acepta una serie de parámetros que son utilizados para entregar al *shell* los resultados solicitados, que pudieran ser latencia, tasa de paquetes recibidos, energía residual de las baterías, tramas recibidas sin interferencia, etc.

Desde la versión 3.0 Castalia cuenta con el *script* CastaliaPlot para la construcción de gráficas a través de la información recopilada en el archivo de salida. Se apoya de la herramienta gnuplot de los sistemas Linux y permite plotear gráficos superpuestos, histogramas, personalizar colores, establecer una relación de aspecto personalizada, establecer leyendas, entre otras opciones.

2.3.3 Programación de módulos en Castalia

Castalia provee la infraestructura necesaria para crear nuevos algoritmos, pero ciertas reglas deben seguirse para su integración con el programa de simulación. Los nuevos módulos deben acoplarse a la jerarquía ya existente e implementar las interfaces necesarias. Asimismo, las clases que brindan funcionalidad a los nuevos módulos deben

heredar de clases virtuales base que implementan los comportamientos básicos esperados de cualquier módulo de su tipo. Por ejemplo, la clase base de enrutamiento VirtualRouting provee funciones para encapsulado/desencapsulado de paquetes, mapeo de direcciones de red a direcciones MAC y manejo de duplicados. De esta manera se asegura la continua innovación del simulador a través de código propuesto por terceras fuentes, y una adaptación más sencilla del código ante las nuevas versiones del simulador.

Por defecto el simulador no incluye módulos de protocolos de enrutamiento utilizados en redes de sensores. La razón de esto es que Castalia inicialmente fue diseñado para proveer un ambiente de simulación lo más real posible, por lo que su principal fuerza se centra en las capas MAC y física.

2.4 Programación del algoritmo FROMS

FROMS es un algoritmo de enrutamiento inteligente y *multicast* para redes de sensores inalámbricos. Sin embargo, en ninguno de los simuladores de código libre encontrados existe la implementación de un protocolo de enrutamiento *multicast* para redes de sensores, por lo que en las simulaciones solamente se utiliza un sumidero. Esto trae como consecuencia algunas consideraciones previas a la implementación del módulo.

Dado que existe un único sumidero en la red, no es necesario dividir los procedimientos de toma de decisiones en acciones y sub-acciones. Entonces, partiendo de la ecuación 2.1 y sustituyendo $|D_i| = 1$ se obtiene que:

$$Q(a_i) = \left(\sum_{d \in D_i} saltos_d^{n_i} \right) \tag{2.11}$$

Esto significa que si un vecino indica que puede acceder al sumidero en n saltos, entonces el valor Q inicial para ese vecino será precisamente n. Consecuentemente, las tablas de enrutamiento se reducen, ya que solamente necesitan almacenar información relativa a cada vecino.

La implementación de FROMS consiste en la definición de un nuevo módulo de enrutamiento que modele el comportamiento del protocolo. Para ello, se debe definir la tabla de enrutamiento, los formatos de paquetes, y el algoritmo de enrutamiento. La figura 2.7 muestra la relación entre el módulo principal FromsRouting y sus clases auxiliares.

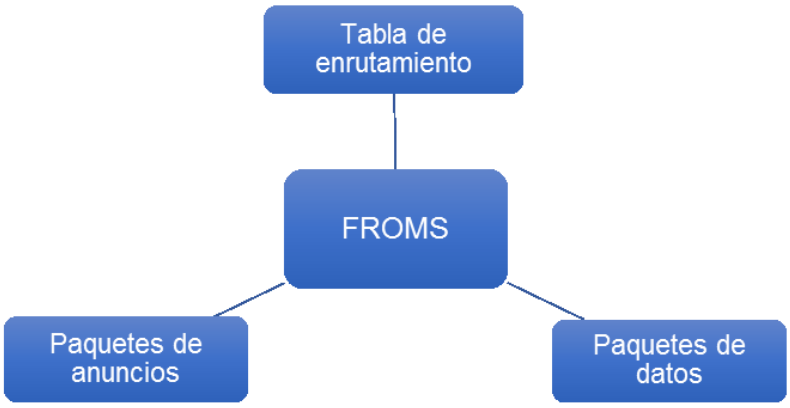


Figura 2.7. Relación entre las clases programadas.

2.4.1 Implementación del módulo de enrutamiento

El módulo de enrutamiento que modela el algoritmo FROMS se denomina FromsRouting. El primer paso para la codificación es definir la estructura del módulo simple en lenguaje de descripción NED (*Network Description Language*). Los parámetros del módulo son utilizados para inicializar ciertos campos críticos en el código C++. Adicionalmente se definen compuertas para el intercambio de mensajes con los módulos de aplicación y MAC. La estructura general de FromsRouting.ned se presenta en la figura 2.8.

El parámetro tiempoInic representa el tiempo de inicialización del módulo de red, esto es, el instante de tiempo en el que el sumidero anuncia su presencia a través de un *SINK_ANNOUNCE*, que permitirá a los nodos recolectar información de enrutamiento inicial. Adicionalmente, con el parámetro intervaloAnuncios, se indica el intervalo de tiempo entre transmisiones de mensajes *SINK_ANNOUNCE* desde el sumidero. La transmisión periódica de estos mensajes es obligatoria para el manejo de fallas, ya que FROMS no implementa ningún mecanismo de manejo de vecinos mediante mensajes de control.

Los parámetros tpoVidaEntradaSink y tpoVidaEntradaVec establecen un tiempo de vida máximo para las entradas del sumidero y de los vecinos en la tabla de enrutamiento, respectivamente. Una vez expirado ese tiempo, se invalidan las entradas correspondientes en la tabla.

El tiempo de vida de la entrada de un vecino se incrementa en tpoVidaEntradaVec milisegundos cada vez que se recibe información actualizada del vecino. De manera similar, cada vez que se recibe un mensaje *SINK_ANNOUNCE*, se incrementa en tpoVidaEntradaSink segundos el tiempo de vida de las rutas hacia el sumidero, así como la entrada del vecino que entregó el mensaje al nodo. Más adelante, en la definición del comportamiento del módulo en lenguaje C++ se explica con más detalle cómo mediante el uso de temporizadores se maneja la validez de las entradas.

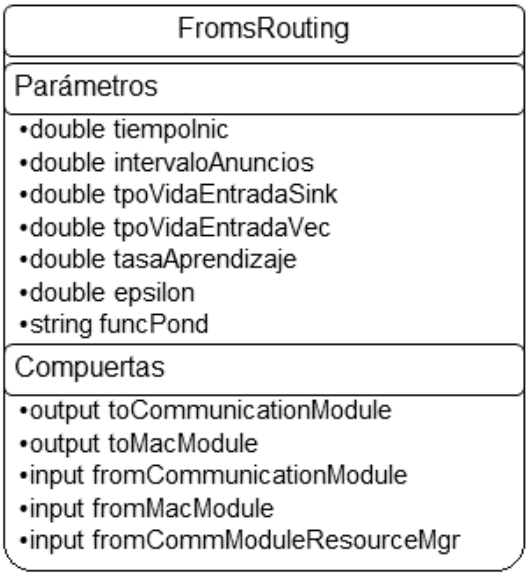


Figura 2.8. Estructura del módulo FromsRouting en Castalia.

Los parámetros tasaAprendizaje y epsilon se corresponden a los parámetros de aprendizaje. La elección adecuada de estos valores es crítica para obtener el mejor desempeño por parte del algoritmo inteligente. Por último, funcPond se corresponde con la función ponderadora utilizada por FROMS para degradar los estimados recibidos por los vecinos de acuerdo a la energía residual.

El archivo FromsRouting.ned describe el módulo de enrutamiento, pero no implementa ninguna funcionalidad. Los parámetros se definen con el objetivo de ser accesibles desde el archivo de configuración de la simulación (omnetpp.ini). Por su parte, las compuertas definen el aspecto del módulo, es decir, las interfaces de entrada/salida para el intercambio de mensajes con otros módulos. Es necesario crear un código en C++ (archivos .h y .cc) responsable de dar vida al módulo de enrutamiento. Este código será el encargado de procesar los mensajes recibidos de otros módulos e implementar el comportamiento del algoritmo FROMS.

2.4.2 Clase TablaEnrutamiento

La tabla de enrutamiento del módulo es pequeña, ya que solamente necesita acumular información relacionada con los vecinos del nodo. Es en esencia una lista de objetos de tipo Entrada, una estructura contenida en el archivo de cabecera, y cuya composición se presenta en la figura 2.9.

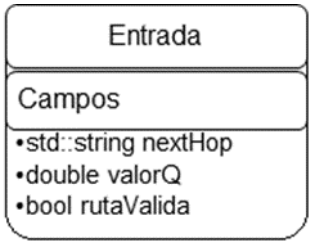


Figura 2.9. Estructura Entrada.

Las entradas están compuesta por tres elementos. El primero de ellos es la dirección del próximo salto, que es alguno de los vecinos del nodo, el segundo elemento es el valor Q asociado al nodo vecino, y el tercero es un indicador de la validez de la ruta. Cada vez que un nodo recibe un paquete, usa los datos de realimentación para calcular el nuevo estimado a partir del estimado actual. Posteriormente, el estimado calculado se introduce en la tabla de enrutamiento, y automáticamente se valida la entrada.

Figura 2.10. Estructura de la clase TablaEnrutamiento.

La clase TablaEnrutamiento provee los mecanismos necesarios para manipular la tabla de enrutamiento utilizada por el algoritmo. El cuerpo de la clase provee los métodos que

permiten manipular los elementos de la tabla de enrutamiento, su composición se puede observar en la figura 2.10. Los métodos son públicos para que sean visibles por el módulo principal, que debe acceder a la tabla y modificar sus campos cuando sea necesario.

2.4.3 Formatos de paquetes

FROMS maneja dos tipos de mensajes: los mensajes de anuncios y los mensajes de datos, cada uno de ellos con un formato diferente. Para representar ambos tipos de mensajes se crean dos archivos de definición de mensajes (.msg): FromsRoutingSinkAnnouncePkt y FromsRoutingDataPkt. Las figuras 2.11 y 2.12 muestran la estructura de cada uno de los paquetes. Los campos de dirección de origen, dirección de destino, y número de secuencia se omiten ya que son heredados de la clase base RoutingPacket.

Paquetes del tipo FromsRoutingSinkAnnouncePkt se envían periódicamente desde el sumidero para anunciar su presencia. La cadena dirSink identifica la dirección del sumidero, que permite a los nodos de la red identificar el destino de los reportes periódicos. El campo cantSaltos contiene la cantidad de nodos por los que transita el paquete. Este valor se incrementa cada vez que un nodo encamina el paquete. De esta manera, el próximo salto es capaz de conocer la distancia en saltos hasta el sumidero desde su ubicación y determinar el mejor vecino para encaminar el próximo paquete de datos.

Cada nodo, antes de reenviar el *SINK_ANNUNCE*, actualiza el campo energResidual. La tupla (cantSaltos, energResidual) permiten al próximo salto degradar el estimado recibido de acuerdo a la energía residual del nodo, dando prioridad a rutas que involucren a nodos de mayor capacidad energética. El campo sinkTimeStamp contiene la estampilla de tiempo, cuya función se explica más adelante.

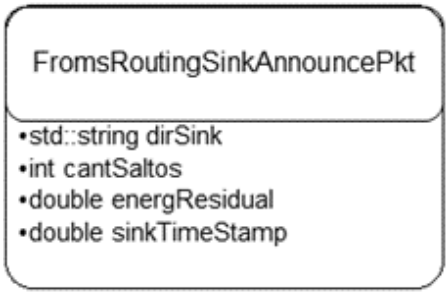


Figura 2.11. Formato del paquete de anuncios del sumidero.

Los paquetes de tipo FromsRoutingDataPkt son, como su nombre indica, los paquetes de datos del algoritmo FROMS. La cadena lastHop especifica la dirección del último nodo por el que transitó el paquete. Este campo, en conjunto con valorQ y energResidual, permiten

a cada nodo actualizar los estimados para cada uno de los vecinos de acuerdo a la regla de actualización del algoritmo.

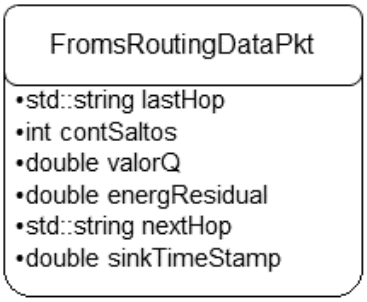


Figura 2.12. Formato del paquete de datos.

Cada vez que un nodo encamina un paquete de datos, reemplaza el valor de lastHop por el de su dirección de red. Acto seguido, busca en su tabla de enrutamiento la mejor entrada válida hacia el destino, actualiza los campos de realimentación del paquete y estampilla de tiempo, incrementa el contador de saltos, y lo envía al módulo MAC para su difusión. El campo sinkTimeStamp almacena una estampilla de tiempo sobre la última vez que el nodo recibió información del sumidero. Si esta estampilla es demasiado vieja, entonces el nodo descarta la información del vecino, puesto que se corre el riesgo de encaminar paquetes hacia un sumidero inactivo.

Los archivos .msg que definen los formatos de paquetes no son utilizados por el programa de simulación. En cambio, son traducidos en tiempo de compilación a clases de C++ que implementarán las funciones de acceso a miembros necesarias. Las clases TablaEnrutamiento y las que definen los formatos de paquete son clases auxiliares ya que realmente no implementan ningún comportamiento específico del algoritmo de enrutamiento. Sin embargo, su implementación es necesaria para que el código que define el módulo sea lo más legible posible.

2.4.4 Implementación del comportamiento del módulo

Todo módulo simple en Castalia tiene una clase en C++ que implementa su comportamiento. Por convención, tiene el mismo nombre que el módulo que describe, en este caso el nombre de la clase es FromsRouting. La figura 2.13 muestra su estructura.

La cadena dirSink almacena la dirección del sumidero. Este campo es necesario debido a que las tablas de enrutamiento reducidas están codificadas por próximo salto, y como consecuencia, no contienen información sobre el destino final. Cada vez que el nodo reciba un anuncio del sumidero, actualizará este valor para confirmar la presencia del mismo.

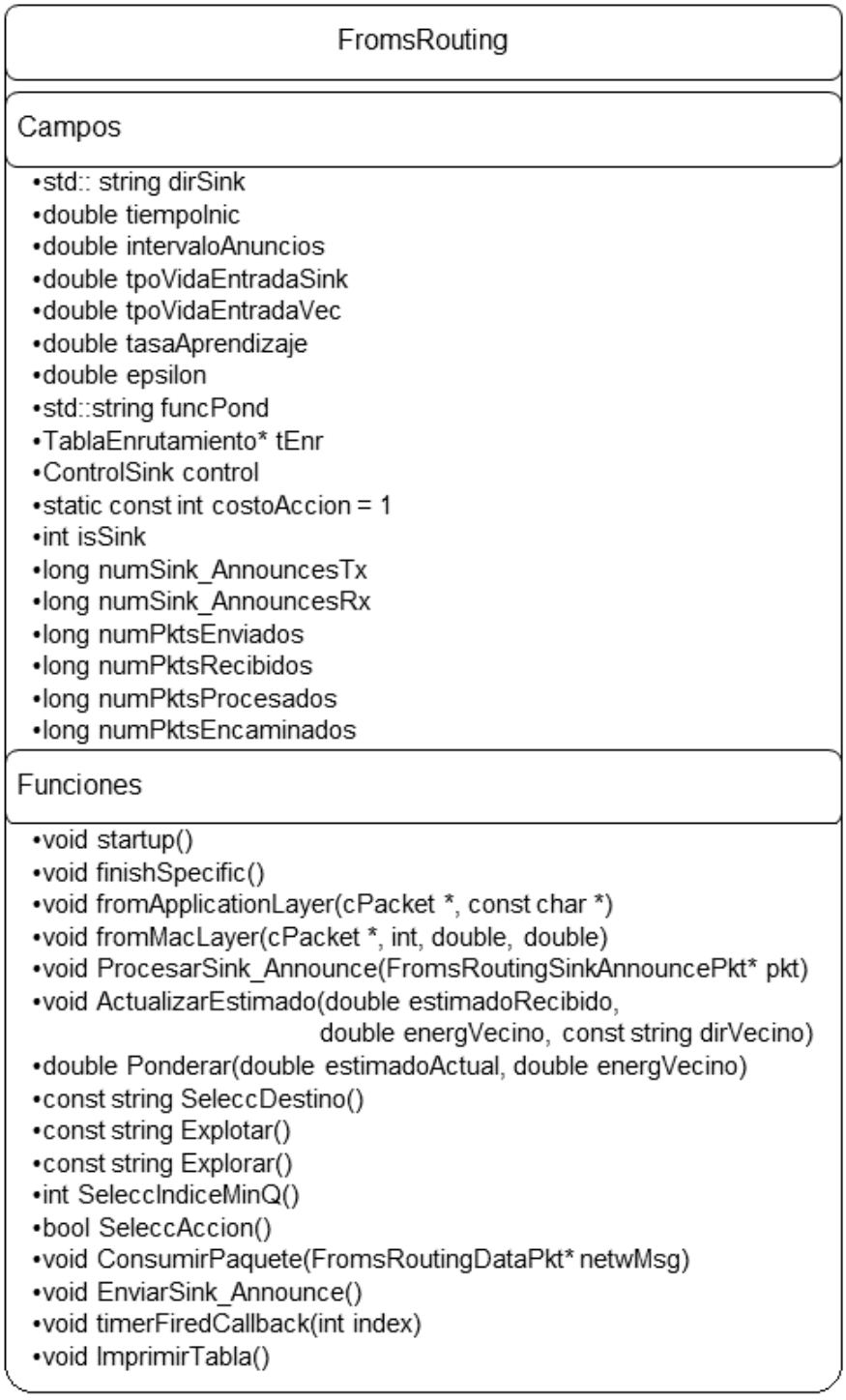


Figura 2.13. Estructura de la clase FromsRouting.

Para determinar el próximo salto para un paquete, el algoritmo puede usar como métrica la distancia en saltos hasta el sumidero, o bien una métrica combinada de saltos y energía residual. Si se usa la métrica simple de saltos, el nodo no necesita apoyarse de ninguna

entidad externa para evaluar los estimados recibidos, puesto que la información necesaria para llenar las tablas de enrutamiento se puede encontrar en los paquetes recibidos. En cambio, si la métrica utilizada es la combinación de saltos y energía residual, la información sobre la energía de la batería del nodo debe ser accesible.

En Castalia, el módulo encargado de gestionar los recursos (estados de la CPU, RAM, batería) se llama resourceManager (ver figura 2.6). Los campos initialEnergy y remainingEnergy representan la energía inicial y residual de la batería en Joules, respectivamente. Ambos campos son privados, por lo que no son accesibles desde otros módulos. Para acceder a estos datos se necesita modificar la clase ResourceManager e implementar al final dos funciones de acceso para ambos campos, como se muestra en la figura 2.14.

Cuando un nodo adjunta información de realimentación a los datos, el indicador de energía que se propaga no es la energía en Joules de la batería, sino el porcentaje de energía residual. Esto se hace con el motivo de adaptarse a los requerimientos de la función ponderadora, que trabaja con indicadores de energía entre 0 y 1. Entonces, es necesario llamar a ambas funciones de acceso y obtener la división entre la energía residual y la energía inicial. Este valor es luego insertado en el campo energResidual del paquete de datos o de anuncios.

```
class ResourceManager: public CastaliaModule {
 private:
    double baselineNodePower;
    double currentNodePower;
    simtime_t timeOfLastCalculation;
    double periodicEnergyCalculationInterval;
    double initialEnergy;
    ⋮
    /*--- Custom class parameters ---*/
    double remainingEnergy;
    ⋮
 public:
    ⋮
    double getInitialEnergy() {return initialEnergy;}
    double getRemainingEnergy() {return remainingEnergy;}
};
```

Figura 2.14. Fragmento de código de ResourceManager.h.

FROMS no usa ningún mecanismo de descubrimiento y manejo de vecinos mediante el intercambio de mensajes de control. Cuando un nodo no recibe información de un vecino o del sumidero durante un determinado tiempo, se invalida la entrada correspondiente dentro de la tabla de enrutamiento. Este comportamiento involucra el uso de temporizadores dentro del módulo de enrutamiento, cada uno de ellos asociado a una entrada dentro la tabla de enrutamiento.

En OMNeT++, los temporizadores se implementan mediante la programación de mensajes hacia el propio módulo. Los mensajes son punteros a la clase base de mensajes cMessage, y pueden programarse para recibirse en el módulo en un tiempo de simulación deseado mediante una llamada al método scheduleAt(simtime_t t, cMessage *msg). Si se desea asociar un temporizador a cada entrada de la tabla, basta con crear un arreglo de punteros a cMessage y programar cada uno de ellos para recibirse en el módulo en el tiempo de simulación deseado.

Sin embargo, esta lógica no puede ser implementada en Castalia, ya que sus desarrolladores redefinen la forma de declaración de temporizadores para ajustarse a los relojes internos de los nodos. En Castalia, la creación de un temporizador se define mediante una llamada al método setTimer(int index, simtime_t time). El parámetro *index* sirve para identificar unívocamente a cada temporizador creado. El segundo parámetro especifica el instante de tiempo en el que se dispara el evento y se ejecutan las acciones correspondientes.

El principal problema es que este mecanismo de creación de temporizadores dificulta la interrelación entre las entradas en la tabla de enrutamiento y su temporizador asociado. Si a la entrada de un vecino i dentro de la tabla de enrutamiento se le asigna un temporizador de índice i, entonces existiría un conflicto con los índices de otros temporizadores usados por el módulo. Por ejemplo, el temporizador de envío de anuncios del sumidero tiene índice 1, y entraría en conflicto con el temporizador del vecino con dirección 1 en la tabla de enrutamiento.

Para enfrentar este problema, se define un *offset* igual a 10 para los índices de los temporizadores. El valor 10 es lo suficientemente grande para la definición de nuevos temporizadores en caso de que en el futuro se vaya a reutilizar el código en la definición de un nuevo módulo. Cada vez que un nodo recibe información sobre un vecino i, cancelará el temporizador anterior en caso de que exista, y creará uno nuevo con índice 10 + i. El

tiempo de activación es igual al tiempo de simulación actual más el tiempo de vida de las entradas de los vecinos.

Los protocolos de capa de enlace de las WSN no implementan confirmaciones de recepción para las tramas de difusión. Para enfrentar este problema, FROMS implementa un mecanismo de confirmación implícito a nivel de capa de red. Cada vez que un nodo envía un paquete, reserva una copia en *buffer* y espera a escuchar la transmisión del vecino. Si durante un tiempo determinado el nodo no percibe que el vecino ha encaminado el paquete, procede a retransmitir la copia en el *buffer*. Los campos de dirección de origen, número de secuencia y último salto se utilizan para identificar unívocamente a cada paquete transmitido. Este comportamiento también es considerado y se definen temporizadores para esta tarea.

Hasta este momento, se han explicado las consideraciones tomadas y mecanismos necesarios para la implementación de FROMS. El resto de los métodos que se incluyen en la clase son propios del simulador, y sus funciones pueden encontrarse en el manual de usuario. Castalia posee varias herramientas para la simulación de redes de sensores inalámbricos, como se expone en la sección 2.3, pero para la realización de las simulaciones en este trabajo, ciertas funcionalidades deben introducirse.

2.5 Modificaciones a Castalia

Castalia recopila información sobre la energía consumida y la energía residual de las baterías de cada nodo al final de las simulaciones. Sin embargo, cuando se tiene una considerable cantidad de nodos, el análisis de los niveles de energía de cada uno por separado se hace muy tedioso. Otro señalamiento es que no existe información disponible sobre el instante de tiempo en que algún nodo ha agotado su energía, ni cuáles son los nodos que llegan a este estado. Para enfrentar estas carencias se deben introducir nuevas funciones dentro del módulo de administración de recursos.

El primer aporte al módulo es la implementación de un histograma de energía residual de las baterías. La información brindada por un histograma es mucho más compacta, y permite determinar con mayor facilidad la distribución de la energía residual entre los nodos de la red. Para la implementación de esta herramienta se agregan dos nuevos parámetros al archivo de descripción del módulo de administración de recursos. El primer parámetro permite ajustar el límite superior en el histograma, mientras que el segundo parámetro determina el número de celdas.

Mediante un apropiado ajuste de ambos parámetros, se puede observar con la precisión deseada la distribución de la energía en la red. Los valores especificados son leídos por el código C++, y la creación del histograma se hace efectiva mediante una llamada al método declareHistogram(const char*, double, double, int) con los parámetros deseados.

Para la recopilación de la información sobre el tiempo de caída de los nodos, así como la identificación de los mismos, se introducen nuevas variables dentro del código C++ del módulo de administración de recursos. Cuando un nodo falla porque ha agotado su energía, se almacena su dirección de red y el tiempo de simulación actual. Estos valores posteriormente son recopilados por Castalia y se presentan al final de la simulación.

Se introduce además un método de estimación del tiempo de vida de la red, que es indispensable para estimar el tiempo de simulación transcurrido al momento de caída del primer nodo, y así determinar un buen tiempo de simulación para los experimentos. Este método fue portado desde Castalia 3.3 hacia la versión 3.2, con algunas modificaciones adicionales. El método original estima el tiempo de caída del primer nodo en días, pero esta escala es demasiado grande para ejecuciones de minutos de duración en tiempo de simulación. Para disminuir el error y obtener un resultado más aproximado se disminuye a minutos la escala dentro del cuerpo del método.

Dado que las estadísticas se desean generar cuando se produce la caída del primer nodo, resulta conveniente detener la simulación cuando ocurre este evento. Esta funcionalidad no existe por defecto en Castalia. Para implementar este comportamiento se introduce un parámetro en el módulo de administración de recursos que activa o desactiva esta opción, y modificaciones menores en el código C++.

En este capítulo se describió la implementación de FROMS en el simulador Castalia. Se comprobó que Castalia provee las herramientas necesarias para realizar simulaciones de redes de sensores inalámbricos. Su estructura modular facilita la integración de nuevos módulos con los ya existentes, y permite personalizar el simulador sin comprometer su integridad.

Se determinó además que REL es un buen candidato para la comparación con FROMS, ya que ambos usan un esquema de encaminamiento consciente de la energía. Además, está disponible el código fuente para Castalia, compartido por los propios desarrolladores del protocolo.

CAPÍTULO 3. SIMULACIONES Y RESULTADOS

Este capítulo está dedicado a evaluar el desempeño del algoritmo de enrutamiento inteligente FROMS mediante el simulador Castalia. El objetivo de esta evaluación es comprobar, si un algoritmo basado en Inteligencia Computacional, ofrece un comportamiento superior con respecto a uno convencional. En este trabajo se realiza la comparación con el algoritmo convencional REL bajo la premisa de que el aprendizaje, en los algoritmos inteligentes, permite tomar decisiones que prolonguen el tiempo de vida de la red.

3.1 Configuración de los parámetros de simulación de Castalia

Antes de la ejecución de una simulación en Castalia es necesario realizar dos pasos. El primero de ellos es el modelado del dispositivo que se va a usar. Aquí se indican cuáles son los módulos que describen el comportamiento del dispositivo, por ejemplo: tipo de aplicación, módulos de comunicación y modelo del transceptor. El segundo paso es especificar los parámetros de simulación. Consiste en describir la topología de la red y especificar el tiempo de simulación y parámetros de cada uno de los módulos utilizados.

3.1.1 Topología de la red

Todas las simulaciones se realizan sobre una red de topología en malla con un aspecto de grilla regular. Este tipo de topologías es muy común en simulaciones para la evaluación de protocolos para redes de sensores inalámbricos [31] [32]. El tamaño de la grilla se fijó en 35 nodos sensores y una estación base, como se muestra en la figura 3.1. El área de despliegue es un terreno de 100 m x 100 m, por los que los nodos están separados a una distancia de 20 m. La estación base se ha ubicado en la esquina superior izquierda, ya que en esta posición se maximiza la cantidad de saltos desde los nodos más alejados. Con esta configuración se puede comparar el rendimiento de los umbrales de REL contra el mecanismo de aprendizaje y funciones de costo de FROMS.

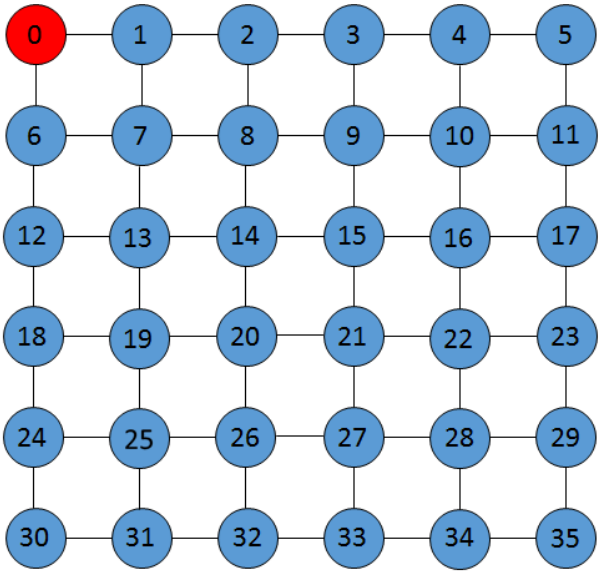


Figura 3.1. Topología de red de prueba.

3.1.2 Modelo del dispositivo

Castalia posee una estructura modular, donde cada nodo sensor se construye mediante la interconexión de módulos para las diferentes capas. Cuando se define un nodo, se debe especificar sus coordenadas y características (protocolos utilizados, transceptor, módulo de aplicación, etc.). La tabla 3.1 recoge los módulos utilizados en la definición de cada uno de los nodos.

Tabla 3.1. Módulos utilizados en la definición de los nodos.

Nombre del módulo	Descripción
ThroughputTest	Aplicación de reportes periódicos
FromsRouting/RelRouting	Protocolo de enrutamiento FROMS/REL
TMAC	Protocolo de subcapa MAC específico para redes de sensores
CC2420	Modelo de transceptor de Texas Instruments

El módulo ThroughputTest se corresponde con una aplicación que genera reportes cada un determinado intervalo de tiempo. Incorpora estadísticas sobre la tasa de recepción, tasa de pérdidas, latencia y cantidad de paquetes recibidos en el nodo designado como sumidero. Dado que tanto REL como FROMS son diseñados para su uso en escenarios de tráfico periódico, este es el módulo de aplicación adecuado para las simulaciones.

El protocolo de subcapa MAC utilizado es T-MAC (*Timeout-MAC*). Este protocolo permite a los nodos sensores activar el transceptor en instantes de tiempo sincronizados, y apagarlos cuando no se detecta actividad en el canal durante un determinado periodo de tiempo. Usa un ciclo útil adaptativo, lo que se traduce en que los nodos consumen más o menos energía en dependencia de la cantidad de tiempo que esté activo su transceptor y la cantidad de paquetes que procesan [33].

La interfaz de red es la CC2420 de Texas Instruments. Este transceptor es compatible con el estándar 802.15.4 y utiliza la banda de 2.4 GHz para comunicaciones de baja potencia y tasa de datos de hasta 250 kbps. La potencia de transmisión es programable y varía desde 0 hasta -25 dBm, como se muestra en la tabla 3.2. La hoja de datos [34] no especifica el alcance de radio para cada uno de los niveles de potencia, por lo que el valor adecuado se debe determinar mediante una simulación. Una potencia de transmisión de -5 dBm es la adecuada para la topología de red de prueba. Niveles mayores provocan la conectividad con nodos más lejanos, mientras que con niveles menores se pierde toda conectividad entre los nodos.

Tabla 3.2. Niveles de potencia de salida del transceptor CC2420 (Fuente: [34]).

Registro TXCTRL	Potencia de salida [dBm]	Consumo [mA]
0xA0FF	0	17.4
0xA0FB	-1	16.5
0xA0F7	-3	15.2
0xA0F3	-5	13.9
0xA0EF	-7	12.5
0xA0EB	-10	11.2
0xA0E7	-15	9.9
0xA0E3	-25	8.5

3.1.3 Parámetros generales de simulación

Para la evaluación de los protocolos se realizan varias simulaciones sobre la topología de red propuesta. Los parámetros generales de simulación se recogen en la tabla 3.3. Estos son los parámetros que se mantienen invariables en todas las simulaciones, y se establecen en el archivo de configuración. Los parámetros para cada uno de los protocolos de enrutamiento varían en cada escenario, por lo que no se listan en la tabla. También se listan

los módulos de cada capa (ver tabla 3.1) con los parámetros anteriormente explicados en el sub-epígrafe 3.1.2.

Tabla 3.3. Parámetros generales de simulación

Parámetro*	Valor
Dimensiones del terreno	100m x 100m
Número de nodos	36
Tiempo de simulación	80 min
Energía inicial de los nodos	100 J
Intervalo entre reportes	10 s
Envío del primer reporte	10 s
Envío del primer anuncio	5 s

El tiempo de simulación es de 80 minutos, un valor lo suficientemente grande para que se produzca la caída del primer nodo y se construya el histograma de energía antes de que se venza el tiempo de simulación. Este valor se determinó mediante el método de estimación del tiempo de vida de la red de Castalia, modificado para las simulaciones en este trabajo.

La energía inicial de los nodos es de 100 Joules. El motivo de elegir este valor es disminuir considerablemente el tiempo de simulación necesario para visualizar los resultados. Dado que tanto REL como FROMS toman decisiones de encaminamiento en base al porcentaje de energía residual de los nodos, no es necesario especificar un valor mucho mayor para la energía de las baterías, lo cual aumentaría considerablemente el tiempo de simulación necesario para que algún nodo agote sus reservas energéticas. De hecho, 100 Joules es el valor especificado en el código de REL proveído por sus desarrolladores para las simulaciones.

El primer anuncio se envía a los 5 segundos de iniciada la simulación, para dar margen a que el protocolo T-MAC sincronice los nodos antes del intercambio de datos. El primer reporte de aplicación se produce a los 10 segundos, cuando ya los nodos han inicializado sus tablas de enrutamiento con la información contenida en los mensajes de anuncios.

3.2 Escenarios de simulación

La tabla 3.4 muestra los cuatro escenarios de simulación considerados. En el primer escenario se varía la función de costo para determinar el impacto de cada una sobre el

tiempo de vida de la red. De acuerdo a los resultados obtenidos, se varía la velocidad de aprendizaje para determinar cuál ofrece el mejor rendimiento en términos de eficiencia energética y tasa de recepción.

Tabla 3.4. Escenarios de simulación.

Escenario	Protocolo	Descripción
1	FROMS	Variación de la función de costo
2	FROMS	Variación de la velocidad de aprendizaje
3	FROMS	Variación de la tasa de exploración
4	FROMS/REL	Comparación de los protocolos

La mejor velocidad de aprendizaje luego se incluye como parámetro en el próximo escenario, donde se varía la tasa de exploración del algoritmo *Q-Learning* utilizado por FROMS. Estos tres pasos permiten determinar los mejores parámetros de aprendizaje para el algoritmo inteligente. Finalmente, en el cuarto escenario se realiza una comparación del rendimiento de REL y FROMS con sus parámetros óptimos, para identificar las ventajas del uso de técnicas inteligentes en el enrutamiento en redes de sensores.

Tabla 3.5. Métricas de evaluación.

Métrica	Resultado deseado
Tiempo de caída del primer nodo	Alto
Distribución de la energía	Grupo compacto
Desviación estándar de la energía residual de las baterías	Baja
Tasa de recepción	Alta

Para la comparación de los protocolos se utilizan las métricas resumidas en la tabla 3.5. Para evaluar la capacidad de extender la vida de la red se consideran tres métricas: distribución de la energía residual en el histograma de energía, tiempo de caída del primer nodo, y la desviación estándar de la energía residual de las baterías. Estas métricas son adoptadas de [25], donde se plantea que el comportamiento deseado es que en cualquier punto dentro del tiempo de vida de la red, el histograma de distribución de la energía debe exhibir un grupo compacto, con todos los nodos teniendo más o menos la misma energía

residual. En las simulaciones se construye el histograma en un solo punto en el tiempo, aquel donde se produce la caída del primer nodo.

La desviación estándar de la energía residual de los nodos es una medida de la dispersión de la energía residual de los nodos. Una desviación estándar baja significa que se logra un buen balance de los gastos energéticos entre los nodos de la red. En las comparaciones también se incluye la tasa de recepción como una medida de la confiabilidad de los protocolos.

3.2.1 Comparación de las funciones de costo

Para observar de manera aislada los resultados obtenidos por cada una de las funciones de costo, se utiliza una política de enrutamiento ávida. En otras palabras, no se realizan acciones de exploración (ϵ = 0), por lo que en todo momento cada nodo utiliza la ruta de mejor estimado en su tabla de enrutamiento. Se fija además una tasa de aprendizaje γ = 1, así cada nodo sustituye inmediatamente los estimados actuales por los recibidos. Todo esto permite ver de manera más clara al final de la simulación la influencia de las funciones de costo sobre los resultados.

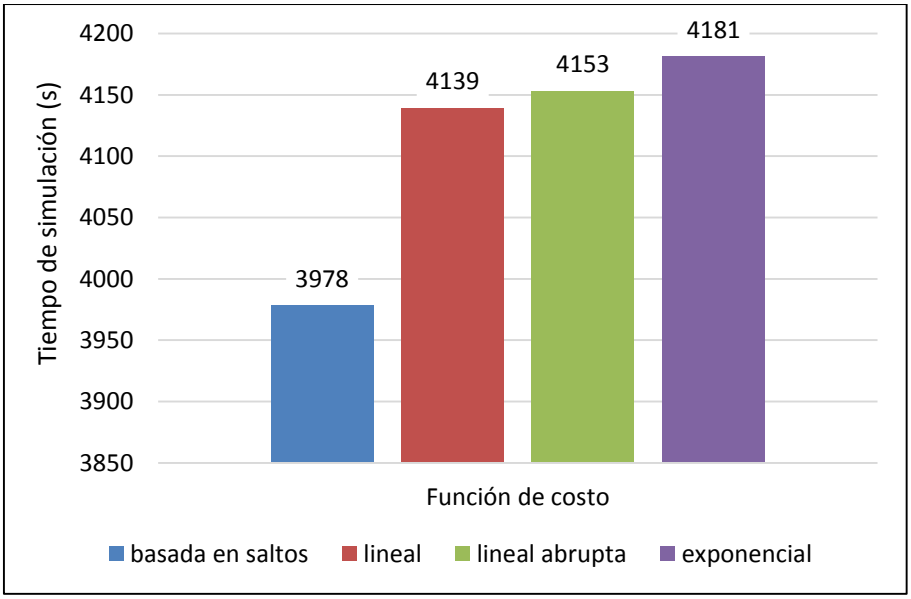


Figura 3.2. Tiempo de vida de la red para cada una de las funciones de costo.

La figura 3.2 muestra el tiempo de caída del primer nodo para cada función de costo. Las funciones de costo que usan una métrica combinada de saltos y energía residual logran aumentar el tiempo de vida de la red entre 161 y 210 segundos con respecto a la función de costo basada en saltos. En este sentido, el mejor resultado se obtiene con el uso de la

función de costo exponencial, que logra un incremento de un 5.1% en el tiempo de vida de la red.

La función basada en saltos selecciona rutas que minimicen la cantidad de saltos requerida para alcanzar el sumidero. Esto se traduce en una disminución de la energía necesaria para alcanzar el sumidero. Sin embargo, los nodos en las rutas más cortas agotan su batería en menor tiempo.

Por el otro lado, las funciones dinámicas de costo, que usan una métrica combinada de saltos y energía, favorecen las rutas más cortas con mayor energía residual en los nodos involucrados, dispersando así el gasto de energía entre los nodos de la red. Este comportamiento se puede observar en la desviación estándar de la energía residual de los nodos. Mientras menor es el valor de la desviación estándar, mejor es el balance de energía obtenido por la función de costo.

La tabla 3.6 recoge el decremento de la desviación estándar de la energía residual para cada función de costo dinámica con respecto a la función de costo basada en saltos. El menor valor se logra con la función de costo exponencial, y esta es la razón por la que incrementa el tiempo de vida de la red en mayor medida, en comparación con las otras funciones.

Tabla 3.6 Desviación estándar de la energía residual para cada función de costo.

Función de costo	Decremento de la desviación estándar
Lineal	2.04 %
Lineal abrupta	5.09 %
Exponencial	5.74 %

El estudio realizado en [25] sobre el uso de funciones de costo en FROMS en un ambiente menos realístico como MATLAB, muestra resultados diferentes. El tiempo de caída del primer nodo usando la función de costo exponencial se extiende en un 80%. En dichas simulaciones, los nodos degradan la energía de su batería en una cantidad fija cada vez que transmiten un paquete, y no se considera el consumo de energía por la escucha pasiva del canal.

En un ambiente más realístico como Castalia, y corriendo el algoritmo de enrutamiento sobre un protocolo MAC real, los resultados no son tan buenos. La razón detrás de esto es

que en Castalia, y en la redes de sensores reales, los nodos consumen energía de acuerdo al estado y tiempo de actividad del módulo de radio, y no precisamente sobre la cantidad de paquetes transmitidos. Por lo tanto, debido a que cada nodo escucha las transmisiones de todos sus vecinos, se consume energía aun cuando no se interviene en el encaminamiento del paquete hacia su destino. Incluso cuando se conmuta hacia rutas alternativas con mayores niveles de energía, la función de costo dinámica selecciona rutas con estimados similares a la anterior.

3.2.2 Selección de la velocidad de aprendizaje

La velocidad de aprendizaje es una constante que determina el tratamiento que se le da a los estimados recibidos. Velocidades de aprendizaje altas dan mayor prioridad a los estimados recibidos sobre los almacenados. Por el otro lado, bajas velocidades de aprendizaje priorizan los estimados acumulados sobre los recibidos. La selección del valor adecuado depende de las fluctuaciones de los valores Q y de los requerimientos de la aplicación.

Para encontrar dicho valor, se prosigue con el uso de una política ávida de selección de rutas, nuevamente con el objetivo de tener información más clara sobre el parámetro que se desea encontrar. En este caso se utiliza la función de costo exponencial, que es la que más prolonga el tiempo de vida de la red.

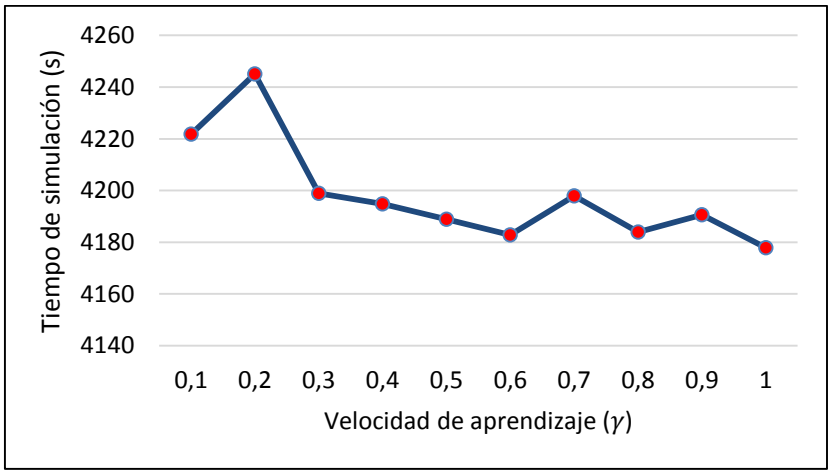


Figura 3.3. Tiempo de vida de la red para diferentes velocidades de aprendizaje.

La figura 3.3 muestra el tiempo de vida de la red en función del tiempo de caída del primer nodo para cada velocidad de aprendizaje γ. Como se puede apreciar, el mejor resultado se obtiene para una velocidad de aprendizaje $\gamma = 0.2$. Esto se debe a que los estimados

recibidos no varían mucho en el tiempo, ya que la energía de los nodos no decae bruscamente. Por tanto, con bajas velocidades de aprendizaje se logra una mejor adaptación ante las fluctuaciones de los valores Q.

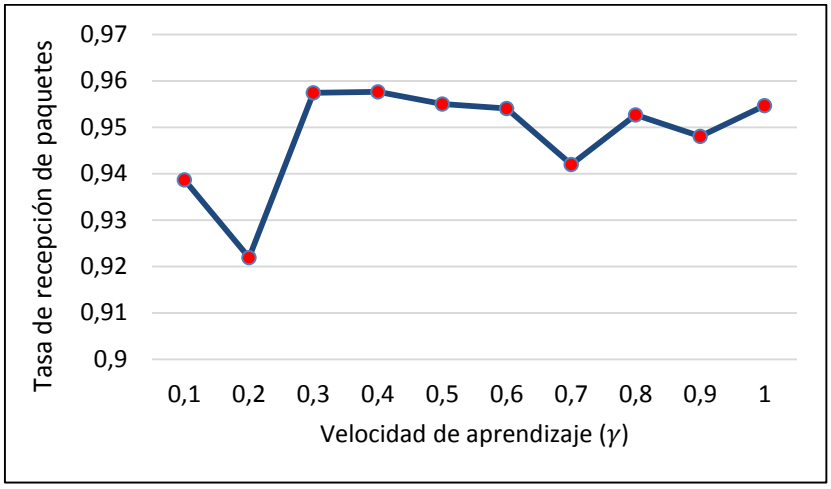


Figura 3.4. Tasa de recepción de paquetes para diferentes velocidades de aprendizaje.

La figura 3.4 recoge la tasa de recepción de paquetes en el sumidero para cada velocidad de aprendizaje. En todos los casos se obtienen valores por encima del 92%, que es un buen valor si se considera que la mayoría de las aplicaciones de redes de sensores requieren tasas de recepción de al menos 80 % [29].

En el anexo I se puede observar la influencia de la velocidad de aprendizaje sobre la latencia. Los requerimientos de latencia y tasa de recepción de paquetes varían de acuerdo a la aplicación, pero el tiempo de vida de la red es un parámetro crítico en toda WSN. Por lo tanto, para los escenarios siguientes se utiliza una tasa de aprendizaje de 0.2, ya que es la que más extiende el tiempo de caída del primer nodo.

3.2.3 Selección de la tasa de exploración

La tasa de exploración (ϵ) define la probabilidad de realizar acciones exploratorias en búsqueda de rutas óptimas. Tasas de exploración bajas provocan que el algoritmo explote con más frecuencia su mejor ruta en la tabla de enrutamiento. Lo opuesto sucede para altas tasas de exploración. Para determinar el valor adecuado, se fija la velocidad de aprendizaje en 0.2, y se varía la tasa de exploración desde 0 hasta 0.3, en pasos de 0.1. En todas las repeticiones de la simulación se usa la función de costo exponencial.

La tasa de exploración tiene un fuerte impacto en la tasa de recepción de paquetes, como se muestra en la figura 3.5. Para $\epsilon > 0$, la tasa de recepción cae bruscamente, coincidiendo

en este aspecto con los resultados obtenidos por los creadores de FROMS en sus simulaciones en MATLAB. Esto se debe a un doble comportamiento exploratorio: la exploración implícita de la función de costo y la técnica de exploración ϵ-*greedy.*

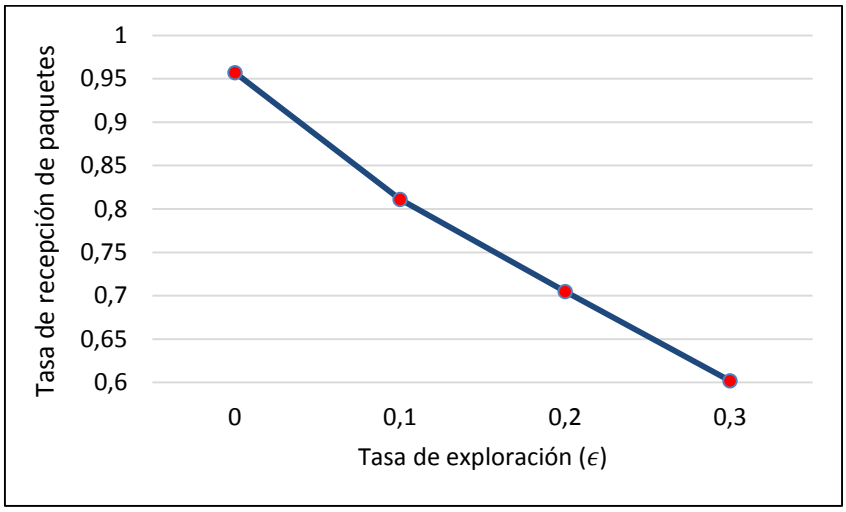


Figura 3.5. Tasa de recepción de paquetes para diferentes tasas de exploración.

Este comportamiento se puede apreciar con más detalle en la figura 3.6, obtenida mediante el *script* de gráficos de Castalia. A medida que aumenta la tasa de exploración, y los nodos utilizan rutas sub-óptimas con más frecuencia, se produce un incremento en las pérdidas de paquetes procedentes de los nodos más alejados.

Esta tendencia es consecuencia de que las acciones de exploración no están limitadas a un conjunto de vecinos, y todos tienen la misma probabilidad de ser elegidos como próximo salto en el camino del paquete. Por lo tanto, es posible que se encaminen paquetes hacia nodos más alejados del sumidero. El número de nodos intermedios tomando acciones sub-óptimas aleatorias aumenta con la distancia, lo que provoca que los paquetes deambulen por la red, tomando rutas más largas y potencialmente malas, e incrementando así las pérdidas.

Antes de introducir el valor Q del vecino en la regla de actualización, la función de costo degrada el estimado recibido de acuerdo al nivel de energía residual de la batería. Por lo tanto, en todo momento se tiene un estimado casi exacto del costo de encaminar un paquete hacia cualquiera de los nodos en la vecindad, por lo cual un mecanismo de exploración explícito se hace redundante.

Visto de otra manera, cuando se usan funciones dinámicas como la función de costo exponencial, cuyos valores varían en el tiempo, incluso con una tasa de exploración nula

se pueden aprender las mejores rutas. Esto es posible porque los estimados recibidos también varían, y con ellos los valores Q de cada ruta almacenada. En otras palabras, incluso cuando la tasa de exploración se fija en 0, FROMS está implícitamente explorando la red, por lo que otro mecanismo de exploración externo se hace indeseable.

De acuerdo a los resultados obtenidos en este escenario, los mejores parámetros para el aprendizaje de FROMS en la topología de red de prueba son $\gamma = 0.2$ y $\epsilon = 0$, y la función de costo que mejores resultados obtiene en cuanto al tiempo de vida de la red es la función de costo exponencial.

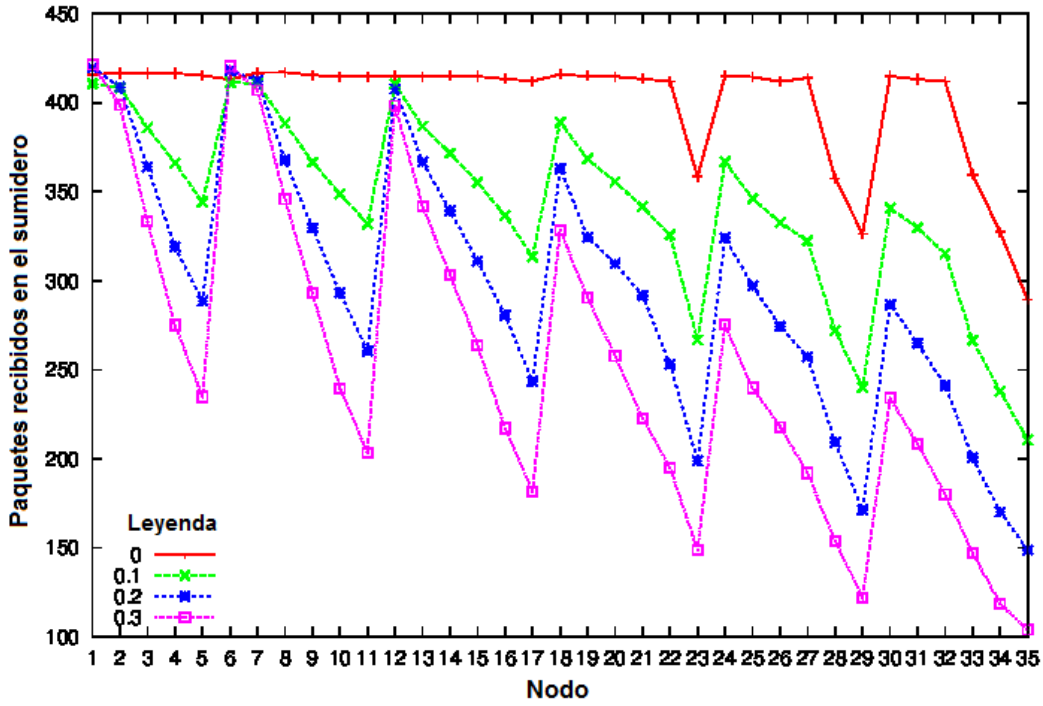


Figura 3.6. Paquetes recibidos en el sumidero por cada nodo en la red, resultados para diferentes tasas de exploración.

3.2.4 Comparación de FROMS y REL

Para identificar las potencialidades de FROMS se realiza una comparación con el protocolo REL. Para ello se corren dos simulaciones por separado, cada una utilizando uno de los protocolos. Las métricas consideradas para la comparación son las de la tabla 3.5.

La tabla 3.7 muestra los resultados obtenidos por cada protocolo en cuanto al tiempo de vida de la red, tasa de recepción de paquetes y desviación estándar de la energía residual. La tasa de recepción de paquetes en el sumidero alcanzada por FROMS es ligeramente menor que la de REL, esto se debe a que las transmisiones se realizan por difusión, por lo

que aumenta el número de colisiones. Para disminuir las pérdidas de paquetes, FROMS incrementa un mecanismo de confirmación de recepción pasivo, ya que las confirmaciones de recepción para tramas de difusión no están definidas en T-MAC, pero esto no es suficiente para alcanzar los resultados alcanzados por REL.

REL, por su parte, utiliza envíos *unicast* para sus paquetes de datos. Esto le permite beneficiarse del *handshake Request-To-Send* (RTS) / *Clear-To-Send* (CTS), reduciendo así posibles colisiones. También se apoya de mensajes de confirmación de recepción a nivel de enlace de datos.

Tabla 3.7. Resultados obtenidos por cada protocolo.

Protocolo	Tasa de recepción	Tiempo de caída del primer nodo	Desviación estándar de la energía residual
FROMS	95.73 %	4206 segundos	5.47 J
REL	97.54 %	3585 segundos	2.23 J

FROMS logra incrementar el tiempo de vida de la red en un 17 % con respecto a REL, en parte debido a un menor gasto de energía por la transmisión de mensajes de control para el mantenimiento de rutas. Cuando se usa el protocolo REL, cada vez que algún nodo excede el umbral de energía residual E_{th}, informa a sus vecinos mediante un mensaje RADV sobre su estado de energía, de manera que éstos evalúen su inclusión en la ruta hacia el sumidero. Este esquema permite el cambio hacia rutas más largas, evitando los nodos con bajos niveles de energía residual, pero el envío de los mensajes RADV también acarrea un costo energético.

Por el otro lado, FROMS se apoya de la realimentación adjunta a los datos para informarse sobre los estimados y estado energético de sus vecinos. Debido a que las transmisiones se realizan por difusión a nivel de enlace de datos, cada nodo puede escuchar las transmisiones de sus vecinos, por lo que no se requieren gastos adicionales para la transmisión de información de control.

Otra causa de la eficiencia de FROMS es el algoritmo inteligente subyacente. La velocidad de aprendizaje es capaz de suavizar los estimados recibidos y usar por un tiempo mayor la ruta actual antes de conmutar hacia otra. En otras palabras, no reemplaza directamente el estimado actual de la ruta por la realimentación del vecino, como sucede en los protocolos

de enrutamiento convencionales. Este comportamiento permite acumular experiencia, y a largo plazo, aumentar el rendimiento.

La figura 3.7 muestra el histograma de energía en el instante de tiempo en que ha caído el primer nodo. El número de nodos con niveles de energía menores que el 5% casi se duplica cuando se usa el protocolo REL, lo mismo sucede en el intervalo del 5-10 %. Esto es en parte debido a la necesidad de transmitir los RADV de manera separada, que conlleva a un aumento de los gastos energéticos.

La otra razón es que REL depende de que se cumplan varias condiciones antes de conmutar hacia otra ruta alternativa. Incluso cuando la energía residual del próximo salto en la ruta es menor que la de algún otro vecino en una ruta alternativa, el algoritmo no conmuta si se excede el umbral de saltos $HCdiff_{\max\_allow}$, o si la ruta alternativa posee mayor cantidad de enlaces débiles. Aunque esto representa una desventaja con respecto a FROMS, forma parte de la confiabilidad característica de REL.

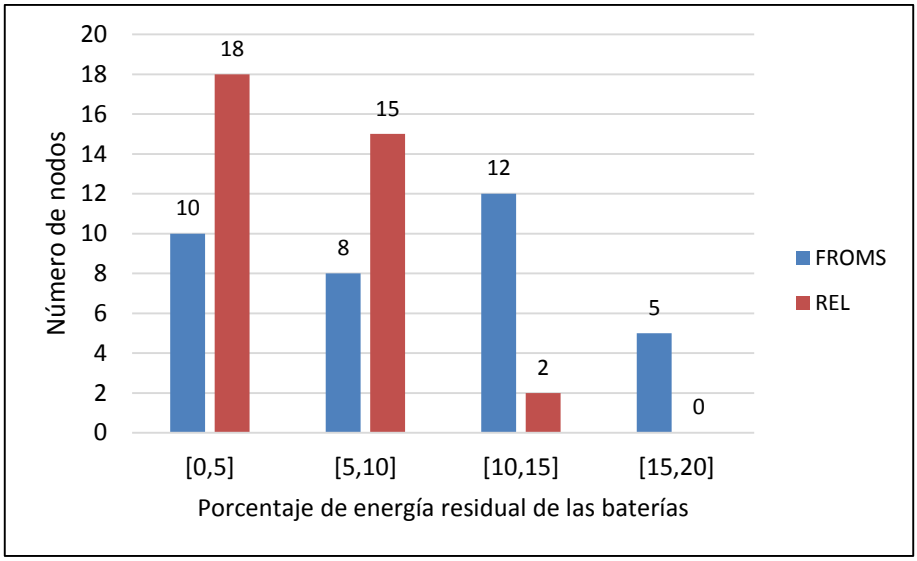


Figura 3.7. Histograma de energía residual de las baterías en el instante de caída del primer nodo.

Si se observa con detalle el histograma de la figura 3.7, REL logra una mejor distribución de los gastos energéticos entre los nodos de la red, casi logrando un grupo compacto entre un 5 % y 10 % de la energía de los nodos. De hecho, la desviación estándar de la energía residual de las baterías obtenida por REL es menor que la alcanzada por FROMS. Sin embargo, se agota la energía de las baterías más rápidamente.

Con esto se muestra la importancia de disminuir al máximo posible los costos de comunicación en una red de sensores inalámbricos, y la capacidad de FROMS de aprender sobre el estado de la red sin gastos adicionales.

3.3 Conclusiones del capítulo

Las funciones de costo proveen un mecanismo de exploración implícito e independiente de la técnica de exploración utilizada. De hecho, eliminan la necesidad de una técnica de exploración externa, que aumenta los gastos de enrutamiento e impacta negativamente en el rendimiento de la red.

FROMS permite obtener beneficios en el ahorro de energía a corto o largo plazo mediante el ajuste de las velocidades de aprendizaje. Cada velocidad de aprendizaje impacta de manera distinta en el tiempo de vida de la red, tasa de recepción de paquetes y latencia.

Mediante la inclusión de la realimentación en los paquetes de datos, se disminuye significativamente el *overhead* de control. De esta manera, disminuye el consumo por transmisiones, a la vez que se aprende una política de enrutamiento consciente de la energía.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- La Inteligencia Computacional introduce mecanismos adaptativos a los algoritmos de enrutamiento en redes de sensores inalámbricos. Estos mecanismos están fundamentados en reglas de aprendizaje capaces de tomar decisiones de enrutamiento óptimas en base a una o varias métricas, aprendizaje que permite tomar las mejores decisiones a largo plazo.
- Dentro de los paradigmas de Inteligencia Computacional, el aprendizaje por refuerzos es el más adecuado para lidiar con problemas distribuidos como el enrutamiento en redes de sensores inalámbricos, debido a su flexibilidad y bajos requerimientos de procesamiento y memoria. Estos factores llevan a la selección del algoritmo de enrutamiento inteligente FROMS para su estudio y simulación.
- Para evaluar el desempeño del algoritmo de enrutamiento FROMS se usa el simulador Castalia, ya que posee las herramientas necesarias para simular redes de sensores inalámbricos en un ambiente realístico. Castalia tiene una arquitectura modular que facilita la implementación y evaluación de nuevos protocolos. Esto permite realizar modificaciones en el módulo de administración de recursos para implementarle el protocolo inteligente FROMS.
- La comparación de FROMS con el algoritmo convencional REL muestra que FROMS es más eficiente en cuanto al consumo de energía, ya que reduce los gastos en el descubrimiento y mantenimiento de rutas, y mediante el uso de funciones dinámicas de costo y una regla de actualización puede seleccionar en todo momento las rutas más cortas hacia el sumidero, a la vez que incrementa el tiempo de vida de la red.
- Los resultados obtenidos sobre la red de prueba muestran que FROMS logra extender el tiempo de vida de la red en un 17% con respecto a REL.

Recomendaciones

- Realizar simulaciones con una mayor cantidad de nodos, lo cual no se pudo lograr debido a los altos requerimientos computacionales.

REFERENCIAS BIBLIOGRÁFICAS

[1] J. Capella, *Redes inalámbricas de sensores: Una nueva arquitectura eficiente y robusta basada en jerarquía dinámica de grupos,* Universidad Politécnica de Valencia, Valencia, 2010.

[2] Zigbee Alliance, *Zigbee Specification,* 2008.

[3] R. Hidalgo y J. I. Moreno, *Routing Design in Wireless Sensor Networks and a Solution for Healthcare Environments, IEEE Latin America Transactions*, vol. 9, No. 3, junio 2011.

[4] A. Mercado, R. B. Figueroa y P. C. Ye, *Redes inalámbricas ad-hoc,* Universidad de Castilla-La Mancha, Castilla-La Mancha, 2010.

[5] M. E. Díaz, *A generic software architecture for portable applications in heterogeneous wireless sensor networks,* Universidad Politécnica de Madrid, Madrid, 2009.

[6] W. Heinzelman, A. Chandrakasan y H. Balakrishnan, «*Energy-Efficient Communication Protocol for Wireless Sensor Networks*,» de *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Hawaii, febrero 2000.

[7] W. Heinzelman, J. Kulik y H. Balakrishnan, «*Adaptive Protocols for Information Dissemination in Wireless Sensor Networks*,» de *Proceedings of the ACM MobiCom'99*, Seattle, enero 1999.

[8] M. R. Modesti, D. O. Tanburi y D. A. Benasulin, *Protocolo de ruteo adaptable para redes inalámbricas de sensores,* Universidad Tecnológica Nacional, Córdoba, 2012.

[9] R. V. Kulkarni, A. Förster y G. Kumar, «*Computacional Intelligence in Wireless Sensor Networks*», *IEEE Communications Surveys and Tutorials*, vol. 13, 2011.

[10] L. Kaelbling, M. Littman y A. Moore, «*Reinforcement Learning: A Survey,*» *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[11] W. H. Andrag, *Reinforcement Learning for Routing in Communications Networks,* Universidad de Stellenbosch, Stellenbosch, 2003.

[12] A. M. Ortiz Torres, Técnicas de enrutamiento inteligentes para redes de sensores inalámbricas, Universidad de Castilla-La Mancha, Castilla-La Mancha, 2011.

[13] J.-H. Chang y L. Tassiulas, «*Maximum Lifetime Routing for Wireless Sensor Networks*,» *IEEE/ACM Transactions On Networking*, Vol. 12, No.4, 2005.

[14] M. Woo y S. Singh, *Power-Aware Routing in Mobile Ad Hoc Networks,* de *Proceedings of the 4th annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*. ACM Press, 1998, pp. 181-190.

[15] A. S. Tanenbaum, *Redes de Computadoras*, Prentice Hall, 2003.

[16] J. Garbarino, *Protocolos para redes inalámbricas de sensores,* Universidad de Buenos Aires, Buenos Aires, 2011.

[17] J. O. M. Alarcón, *Sistema Inteligente de monitoreo para el consumo de energía de redes inalámbricas ad hoc,* Universidad Nacional de Colombia, Bogotá, 2013.

[18] J. Barbancho, C. León, J. Molina y A. Barbancho, «*Giving neurons to sensors. QoS management in wireless sensor networks*,» de *Proc. IEEE Conf. Emerging Technology. Factory Automation ETFA*, Castilla y León, marzo 2006, pp. 594-597.

[19] I. Gupta, D. Riordan y S. Sampalli, «*Cluster-head election using fuzzy logic for wireless sensor networks*,» de *3rd Annual Communications Networks Services Research Conference*, Chicago, marzo 2005.

[20] O. Islam y S. Hussain, «*An Intelligent Multi-hop Routing for Wireless Sensor Networks*,» de *Proc. WI-IAT Workshops Web Intelligence International. Agent Technology*, diciembre 2006.

[21] S. Wazed, A. Bari, A. Jackel y S. Bandyopadyay, «*Genetic algorithm based approach for extending the lifetime of two-tiered sensor networks*,» de *Proc. 2nd Int. Symp. Wireless Pervasive Computing ISWPC*, Bari, 2007.

[22] C. Domínguez, *Algoritmos bioinspirados para el encaminamiento de datos en redes inalámbricas de sensores*, Instituto Politécnico Nacional, México D.F, 2011.

[23] R. S. H. Hernández, *Algoritmo de colonia de hormigas para el enrutamiento en redes de sensores inalámbricos,* Universidad Centroccidental, Barquisimeto, 2011.

[24] J. A. Boyan y M. L. Littman, «*Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach*,» de *Advances in Neural Information Processing Systems, vol. 6*, 1994.

[25] A. Förster y A. L. Murphy, «*Balancing Energy Expenditure in WSNs throgh Reinforcement Learning: A Study*,» de *Proc. 3rd Int. Conf. Intelligent Sensors, Sensor Netw. Inf. Process*, 2007.

[26] [En línea]. Available: http://www.isi.edu/nsnam/ns/. [Último acceso: enero 2014].

[27] [En línea]. Available: http://www.omnetpp.org/. [Último acceso: enero 2014].

[28] [En línea]. Available: http://nsr.bioeng.washington.edu/jsim. [Último acceso: enero 2014].

[29] K. Machado, D. Rosário, E. Cerqueira, A. A. F. Loureiro, A. Neto y J. N. d. Souza, «*A Routing Protocol Based on Energy and Link Quality for Internet of Things Applications*,» *Sensors,* vol. 13, pp. 1942-1964, 2013.

[30] A. Boulis, *Castalia User's Manual,* 2011.

[31] M. Devillé, Y.-A. L. Borgne y A. Nowé, *Reinforcement Learning for Energy Efficient Routing in Wireless Sensor Networks,* Universidad Libre de Bruselas, Bruselas, 2011.

[32] Y. Zhang y Q. Huang, «*A Learning-based Adaptive routing Tree for Wireless Sensor Networks*,» de *Proceedings of the IEEE 3rd Consumer Communications and Networking Conference*, Las Vegas, enero 2006.

[33] J. M. van Dam, *An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks*, Delft University of Technology: Delft, Holanda, 2003.

[34] Texas Instruments, *CC2420 datasheet,* Rev. SWRS041b, marzo del 2007.

GLOSARIO DE TÉRMINOS

ACO-QoSR *Ants Colony Optimization based QoS Routing algorithm*

AE Algoritmo evolutivo

AODV *Ad hoc On Demand Distance Vector*

CDRQ-Routing *Confidence-based Dual Reinforcement Q-Routing*

CQ-Routing *Confidence-based Q-Routing*

CSMA *Carrier Sense Multiple Access*

CTS *Clear-To-Send*

DRQ-Routing *Dual Reinforcement Q-Routing*

DSR *Dynamic Source Routing*

FROMS *Feedback ROuting to Multiple Sinks*

GA-Routing *Genetic Algorithm Routing*

GUI *Graphical User Interface*

IC Inteligencia Computaciona

LEACH *Low Energy Adaptive Clustering Hierarchy*

LQI *Link Quality Indicator*

MAC *Medium Access Control*

MANET *Mobile Ad hoc Network*

MTE *Minimum Total Energy*

Nam *Network Animator*

NED *Network Description Language*

NS-2 Network Simulator 2

nst-AODV *not so tiny Ad hoc On Demand Distance Vector*

OMNeT++ *Objective Modular Networks Testbed*

QoS *Quality of Service*

RADV *Route Advisor*

REL *Routing by Energy and Link Quality*

RREP *Route Reply*

RREQ *Route Request*

RSSI *Received Signal Strength Indication*

RTS *Request-To-Send*

SIR *Sensor Intelligence Routing*

SNR *Signal-Noise-Ratio*

SOM *Self-Organized Map*

SPIN *Sensor Protocol for Information via Negotiation*

TDMA *Time Division Multiple Access*

T-MAC *Timeout-MAC*

Two-Tier GA *Two-Tier Genetic Algorithm*

USB *Universal Serial Bus*

WSN *Wireless Sensor Network*

ANEXOS

Anexo I. Histograma de latencia para velocidades de aprendizaje entre 0.1 y 1. Los requerimientos de latencia y tasa de recepción vienen dadas por la aplicación para la cual se diseña la WSN.