RDiPro

Algoritmo de diseminación de mensajes a través de RSSI

P.G. Lora
University of Seville
Escuela Técnica Superior de Ingeniería
Avda. Camino de los Descubrimientos, s/n, 41092 Sevilla, Spain
pedgutlor@alum.us.es

G.R.Flores
University of Seville
Escuela Técnica Superior de Ingeniería
Avda. Camino de los Descubrimientos, s/n, 41092 Sevilla, Spain
gabrodflo@alum.us.es

***Abstract*— RDiPro (*RSSI* + *Distance* + *Probability*) es un algoritmo de diseminación de mensajes para redes inalámbrica de sensores. Bajo el uso de la técnica de inundación de red (*flooding*) para hacer llegar un mensaje de un nodo a toda la red, RDiPro ejecuta un reenvío masivo con ciertas limitaciones, donde se utiliza la potencia de la señal recibida para calcular la distancia y usar esta para, a través de una función de probabilidad, decider si el nodo retransmite el paquete recibido. El algoritmo ha sido probado bajo simulación sometiendolo multiples casos, variando la densidad de nodos, el tráfico de red y el tamaño de los mensajes transmitidos, dando como resultado una cierta mejoría sobre el algoritmo sin restricciones en algunos aspectos de estudio.**

Keywords—WSN; Flooding; RSSI; Distance; Probability

I. INTRODUCCIÓN

Una red de sensores inalámbricos o WSN (*Wireless Sensor Network*) consiste en una red de dispositivos autónomos equipados con sensores preparados para medir una magnitud física y que poseen la capacidad de comunicarse entre ellos de forma inalámbrica. Basadas en un funcionamiento descentralizado, las WSN no requieren una estructura predeterminada, así como ninguna topología fija, dado que esta puede cambiar muy frecuentemente según el estado de la red. Así, la red será un conjunto de nodos sensores localizados aleatoriamente de forma muy densa. Estos suelen usar comunicaciones tipo *broadcast* (transmisión de datos recibido por todos los nodos dentro de su alcance). Entre las características más comunes, destaca la búsqueda constante de eficiencia energética, ya que los nodos sensores están muy limitados en consumo; y auto-organización de la red, debido a la gran cantidad de nodos presentes.

Frente a las características mencionadas, nace la necesidad de evaluar el funcionamiento de la red bajo distintos escenarios, donde la densidad de la red, la distribución de los nodos, el número y el tamaño de los mensajes marcarán los resultados de la evaluación. En esta investigación, por medio de simulaciones, se obtendrá una aproximación de la implementación real de una WSN en la que se ha puesto a prueba un algoritmo de *diseminación de mensajes*, cuyo fin es reducir la redundancia de los mensajes de la red. Este análisis da en términos de números de transmisiones totales de la red y número de nodos que han recibido la transmisión.

La diseminación de mensajes es una técnica muy utilizada en WSN, en la cuales cada vez que un nodo recibe un paquete por primera vez, lo retransmite a todos sus vecinos. Este mecanismo tiene varias deficiencias que limitan la escalabilidad debido al consumo excesivo de recursos, tales como ancho de banda, energía y espacio en colas. El problema de optimización relacionado con la diseminación de mensajes, tiene por objetivo reducir el número de transmisiones que se hacen para entregar un paquete a todos los nodos de la red. Para lograr esta tarea, es necesario encontrar un conjunto de nodos mínimo que abarquen todo el espacio de la red, y donde únicamente los nodos pertenecientes a este conjunto sean los que retransmitan el mensaje.

El algoritmo que se expone en este documento trata de cumplir este cometido a través de una retransmisión de paquetes basada en *flooding* o inundación de la red. Cada nodo que ejecute RDiPro rellenará una tabla con sus vecinos (aquellos nodos con los que tenga cobertura) a modo de establecimiento de la red. Una vez pasada la fase de reconocimiento del vecindario, cada nodo tendrá como información el identificador de cada nodo vecino y la distancia que los separa. Cuando un nodo transmite un paquete, este incluye en su cabecera un parámetro que indica el valor de la distancia a la que se encuentra el nodo más lejano de entre todos sus vecinos. Este dato será usado para calcular el valor de la función de probabilidad que decidirá si el paquete será o no retransmitido. Esta distancia es calculada según el valor RSSI que se reciba del nodo fuente, es decir, del nivel de potencia de la señal recibida.

En los siguientes apartados explicamos otros mecanismos de diseminación de mensajes (II) similares a RDiPro que emplean herramientas parejas a las que se han comentado. En la sección (III) se describe al detalle el algoritmo propuesto, explicando los fundamentos y decisiones que se han tomado a lo largo del desarrollo de la investigación. Tras explicar el funcionamiento, en la sección (IV) se muestran los resultados de las simulaciones que se han realizado tanto sobre RDiPro como en otros algoritmos de diseminación de mensajes, todos bajo las mismas circunstancias. Para finalizar, en la última sección (V), se resumen los resultados de estas pruebas, la funcionalidad del algoritmo y se plantean las posibles mejoras sobre las que trabajar en un futuro, para continuar con la optimización de la idea propuesta.

II. Trabajos Relacionados

RDiPro se caracteriza por una serie de aspectos que lo hacen, en cierto modo, único. Pero no significa que no existan protocolos que de forma separada implementen algunas de las funcionalidades que RDiPro utiliza. Es el caso de los tres protocolos propuestos: RAPID [1] (protocolo basado en probabilidad y vecinos), EDAS [2] (basado mayormente en la distancia entre vecinos) y ReMo [3] (utiliza algunos parámetros para determinar la calidad del enlace, como el RSSI, para transmitir en función de estos):

A. *RAPID: Reliable Probabilistic Dissemination in Wireless Ad-Hoc Networks*

Este protocolo usa como método para retransmitir una probabilidad que se determinará en función de dos elementos: el número de vecinos del nodo y la probabilidad de que un nodo reciba un mensaje que previamente fue enviado. Dado que cada nodo necesita conocer el número de vecinos a un solo salto, cada uno envía de forma periódica *hello messages* (a menos que ya haya enviado otro mensaje). En paralelo, cada nodo periódicamente reenvía a sus vecinos las cabeceras de los mensajes recibidos de otros nodos, lo que se conoce como *gossip*. Esta técnica permite que todo nodo que pierda algún mensaje, pueda pedirlo a sus vecinos. Solo envían cabeceras de mensajes que ya posean. Por tanto, la cabecera de un mensaje que no exista no será diseminado por la red. Además, y siempre que sea posible, un mensaje se adhesionará a otro para reducir el tráfico de la red. Para reducir la posibilidad de que mensajes colisionen (una de las razones de más peso por la que los mensajes se pierden), RAPID usa *jitter*. Por tanto, la pequeña probabilidad de retransmitir y el *jitter* tan pequeño antes de retransmitir implica que RAPID raramente causará colisiones.

Los nodos antes de transmitir añaden el mensaje a una cola, y se ponen a monitorizar a sus vecinos. Si descubre que uno de sus vecinos está retransmitiendo ese mensaje, lo elimina de la cola sin retransmitirlo. De lo contrario, y después de que pase un tiempo aleatorio, retransmitirá con la probabilidad mencionada al principio.

B. *EDAS: Energy and Distance Sware protocol based on SPIN*

EDAS es un protocolo que retransmite mensajes en función de la energía de sus nodos vecinos y la distancia entre ellos. El estado del nodo, el cual considera la energía residual y distancia ideal, se define como el *Node marker* N^m. Cuando un nodo transmite un mensaje de anuncio, los vecinos que reciban ese mensaje independientemente calculan N^m, ya que cada nodo es consciente de su localización (usando un sistema de localización ajeno a este protocolo). Cada nodo puede decidir si participar en la transmisión basándose en el valor de N^m, el cual se define como sigue:

$$N^m(\omega_e, \omega_d) = \omega_e N_e + \omega_d N_d$$

$$\omega_e, \omega_d \geq 0;\ \omega_e + \omega_d = 1$$

donde N_e, N_d son la energía residual de cada nodo y la distancia entre dos nodos. Cada uno de estos atributos se puede modificar con un factor de ajuste ω_e y ω_d. En base a esto, el nodo establecerá un temporizador según el valor de N^m (a mayor N^m, de menor duración será el temporizador). Cuando el temporizador expira, el nodo escucha para comprobar si el canal está libre. Si lo está, transmite el mensaje de anuncio a sus vecinos. Si está ocupado, se mantendrá a la escucha del mensaje.

C. *ReMo: An Energy Efficient Reprogramming Protocol for Mobile Sensor Networks*

En ReMo lo que se busca es que un nodo mantenga de forma periódica a sus vecinos actualizados con una versión actualizada del código que tenga que enviar a difusión y además con información sobre su localización. Cuando un código nuevo se propaga por la red, un nodo necesita escoger de forma óptima al vecino más adecuado del que descargar este código. Para ello cada nodo puede estar en tres estados posibles: ADV (*advertise*), RX (*receive*) o TX (*transmit*). En el estado ADV, un nodo transmite un mensaje de anuncio el cual contiene algunos metadatos y lo hace una cierta probabilidad. Cuando un vecino recibe este anuncio, tiene la posibilidad de enviar un *HReq* (petición a medias) o un *FReq* (petición completa). En el primer caso, el nodo que recibe la petición a medias decidirá si responderle con los datos solicitados o si ignorar la petición. La decisión de un nodo de a quién le mandará las peticiones (ya sea *HReq* o *FReq*) vendrá determinado por el vecino que tiene el mayor potencial para transmitir esa información, así como suficiente calidad de enlace (principalmente será el RSSI de la señal). Para ello se calcula una probabilidad que depende de estos dos elementos y se coge al nodo que resulte en la mayor probabilidad. Además, antes de enviar la petición pasará a estado RX y comprobará si la calidad del enlace es superior a un umbral determinado, en ese caso enviará una *FReq*, en caso contrario una *HReq*. Cuando un nodo recibe una petición pasa al estado TX y transmite todos los paquetes necesarios.

III. Algoritmo Propuesto

El algoritmo propuesto basa su funcionalidad en dos aspectos fundamentales para obtener una probabilidad de retransmisión tras recibir un mensaje: de forma indirecta, en el RSSI (el cual indica con qué potencia se ha recibido un mensaje) y, de forma directa, en la distancia entre vecinos. RDiPro pretende solventar las carencias del básico protocolo de inundación o *flooding*, y para ello el objetivo es el de disminuir el número de retransmisiones que se hacen a lo largo de la red. De forma general, el algoritmo hace uso de la distancia entre los vecinos que se encuentran a un salto para, en base a una ponderación, calcular con qué probabilidad el nodo que acaba de recibir el mensaje va a retransmitir el mensaje. La forma en que esta distancia es calculada implica hacer uso del RSSI de los mensajes recibidos.

Previo a la puesta en marcha del algoritmo, se requiere de una fase de establecimiento de vecinos en la cual se obtendrá la distancia relativa con los mismos. Se explica de forma breve el método utilizado para obtener esta tabla de vecinos: al poner los nodos en marcha, estos empezarán a transmitir mensajes en difusión los cuales serán escuchados por todos los nodos que se encuentren a su alcance. Cada nodo tendrá definida una

estructura *neighbor* que contendrá la siguiente información sobre el vecino:

- La dirección Rime del vecino
- El último RSSI y el último estado de la calidad del canal con el vecino
- El último número de secuencia que se intercambió con este nodo
- La distancia con el nodo

La forma en que la distancia es calculada se hace a partir del RSSI del mensaje de difusión. Por defecto, los nodos tienen definida una potencia de transmisión. Usando esto y el RSSI se puede obtener una distancia relativa con la siguiente operación:

$$d = P_{TX} — RSSI \quad (1)$$

Cada vecino, junto con estos datos pasan a ser almacenados en una lista de vecinos.

Una vez generados los vecinos, es posible comenzar a enviar mensajes en difusión empleando el protocolo RDiPro. Cada mensaje llevará una cabecera que contendrá la siguiente información:

- Número de secuencia
- Dirección del nodo origen
- Saltos (número de nodos por los que ha pasado)
- Distancia del vecino más lejano del nodo que envía el mensaje

Por otro lado, cada nodo abrirá una conexión de tipo *rdipro* que contendrá la siguiente información (relevante):

- La dirección del último nodo que le envió un mensaje
- El último número secuencia del mensaje del último nodo que le envió un mensaje
- La distancia del vecino que se encuentra más lejos del nodo

Un nodo origen comenzará la transmisión enviando un mensaje a sus vecinos de un salto. Al recibir los mensajes, mirará que el número de secuencia que incluye la cabecera sea superior al que tiene almacenado como información de conexión, en caso de que el mensaje provenga del mismo nodo origen que envió el mensaje anterior, o bien comprueba que es un nuevo nodo el que está enviando el mensaje (básicamente está comprobando si no es un mensaje que ya ha recibido con anterioridad, pues en tal caso lo descartará). Cumplidas una de estas condiciones, lo siguiente a tener en cuenta es que el número de saltos (que se incluye en la cabecera) del paquete sea inferior a un número de saltos máximos predefinido por el usuario. En caso de que lo supere, el mensaje se descarta y no es retransmitido (se hace con el fin de evitar que el mensaje esté dando vueltas por la red continuamente).

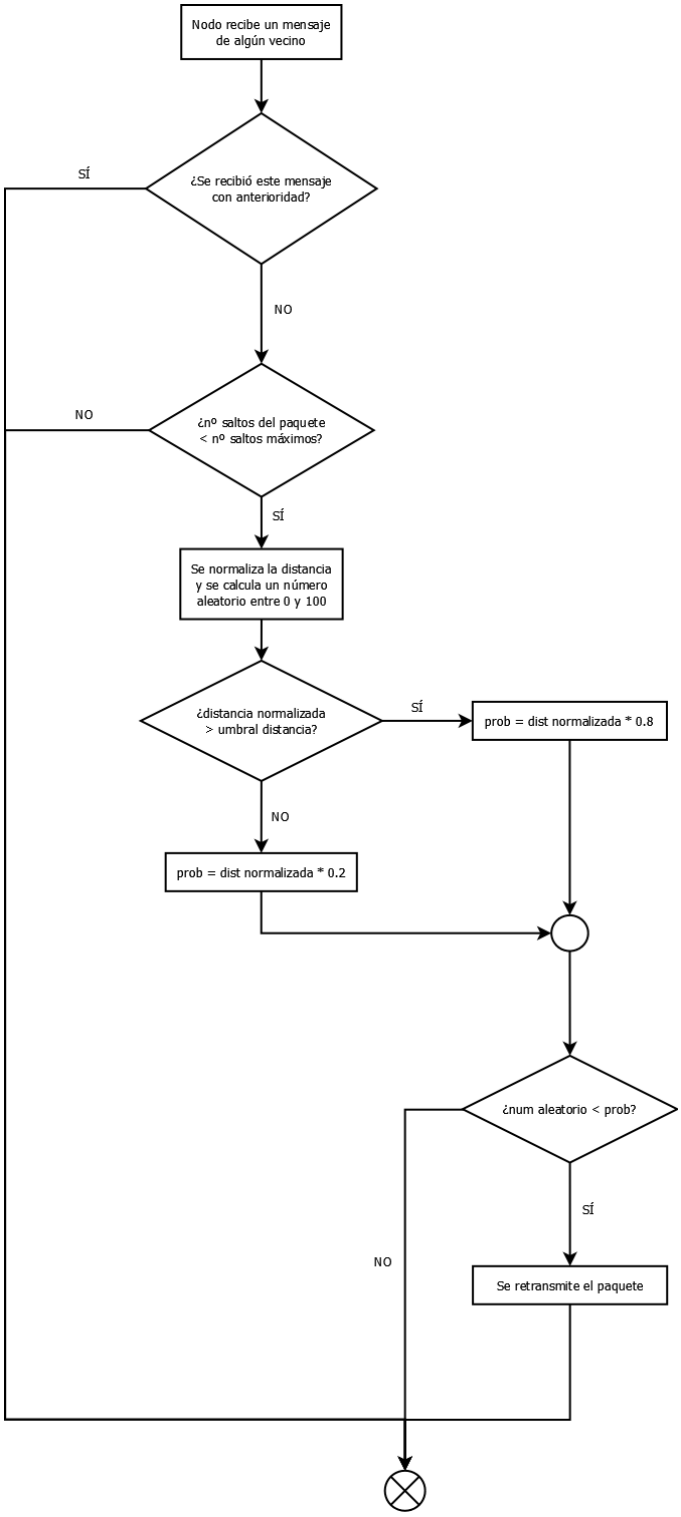

Fig. 1. Diagrama de flujo del algoritmo RDiPro

El siguiente paso tras confirmar que el número de saltos es inferior al establecido como máximo consiste en una serie de operaciones para determinar con qué probabilidad se va a transmitir:

1. Se normaliza la distancia de forma que:

$$d_{norm} = 100 \text{ x } (d_{nodo\ receptor}/d_{vecino\ más\ lejano\ del\ transmisor}) \quad (2)$$

2. Se calcula un número aleatorio entre 0 y 100
3. Se comprueba si la distancia normalizada es mayor que un umbral (configurable por el usuario, siendo 80 el valor por defecto):
    a. En caso de serlo, la distancia normalizada quedará multiplicada por 0.8 (se puede definir por el usuario)
    b. En caso de no serlo, la distancia normalizada quedará multiplicada por 0.2 (se puede definir por el usuario)

De esta forma, estamos haciendo más probable que el nodo que se encuentra más lejos del nodo transmisor tenga más probabilidades de retransmitir (al estar más lejos es probable que tenga a su alcance a muchos más nodos que otro nodo cercano al transmisor ya que podrían incluso ser sus propios vecinos).

Una vez obtenida la probabilidad con que retransmitirán, se compara el número aleatorio con dicha probabilidad. Si es menor, el nodo retransmitirá, si es superior el paquete se descarta.

Así por ejemplo si tenemos una posibilidad de 75 de retransmitir y nuestro número aleatorio es menor de 75, el nodo retransmitirá el mensaje.

Los nodos conformen van recibiendo el mensaje van actualizando la distancia del vecino que tienen más lejos de la cabecera del paquete que se está retransmitiendo.

IV. Resultados de Simulación

Para verificar la funcionalidad de este mecanismo se ha programado este algoritmo bajo el sistema operativo Contiki y probado con el simulador Cooja, que nos permite reproducir una red definiendo muchos de sus parámetros. En concreto, los parámetros con los que se han realizado las pruebas de esta sección se recogen en la siguiente tabla:

TABLE I. DATOS DE SIMULACIÓN

Parámetro de simulación	***Valor [Unidades]***
Tamaño del escenario	210 x 210 [m]
Tipo de nodos	Z1
Radio de cobertura	50 [m]
Número de nodos	25, 50, 75, 100, 125, 150
Distrobución de los nodos	Lineal
Número de mensajes	1, 5, 10, 15, 20
Tamaño de los mensajes	5, 10, 15, 20 [bytes]

Con estos valores, se han realizado diversas pruebas tanto al algoritmo del documento, RDiPro, como a su predecesor flooding. Estas pruebas consisten en plantear una red de nodos distribuidos de manera uniforme sobre un escenario de tamaño fijo, donde el cual se añadirán diferentes números de nodos para variar la densidad de estos y ver cómo funcionan los algoritmos en estas circunstancias. Así conseguimos, además, mayor o menor redundancia en la red, que es el problema que tratamos solventar.

Además de estas simulaciones, también se han realizado unos códigos en Python, a través de los cuales obtenemos los datos numéricos de los resultados. Estos funcionan recogiendo las salidas que imprimen por consola los nodos cada vez que efectúan una acción, como enviar, recibir o retransmitir. Las figuras que se muestran a continuación, también generadas con el mismo lenguaje de programación, son producto de los datos que se recogen en los archivos de texto exportados de las simulaciones y que son procesados por estos programas.

Para las comparaciones que mostramos, se han tenido en cuenta el número de mensajes pertenecientes a la difusión de los paquetes de información hábiles, aquellos usados para compartir y formar la tabla de vecinos no se han contabilizado, ya que este número es de un orden mucho mayor a los que tratamos en este apartado. Hay que decir que estos datos han sido calculados, siendo desde ~500 mensajes (para una red de 25 nodos) hasta ~5000 mensajes (para 150 nodos), sabiendo esto, es razonable pensar que la inclusión de dicha información sólo será relevante cuando la simulación alcance el número de mensajes y retransmisiones de ese mismo orden, donde el protocolo RDiPro ofrece una clara eficiencia siempre y cuando haya una alta densidad de nodos, como veremos en los próximos apartados.

A. Variando el número de nodos

Como se muestra en la tabla de datos, el número de nodos en la simulación varía para cada caso, para, como se ha explicado, hacer un estudio de funcionalidad bajo distintas densidades en la red. Este primer estudio, mostrado en las figuras 2 y 3, refleja cómo se comportan los dos algoritmos bajo el envío de un solo mensaje, en concreto, cuantos nodos de la red reciben el mensaje y cuantas retransmisiones se han realizado en total. En estas pruebas se ha variado la cantidad de nodos de la red desde 25 hasta 150 nodos, en pasos de 25.

En dichas imágenes se aprecia cómo, en general, el número de retransmisiones es menor en RDiPro. Cabe destacar la irregularidad en los resultados, esto se debe a que estos datos han sido sacados de una sola simulación. Sería conveniente realizar estas pruebas de manera repetida y sacar un valor promedio, donde se apreciaría mayor linealidad en la gráfica conforme al número de nodos, y donde RDiPro quedaría en cualquier caso, inferior en número de retransmisiones. Esta suposición se aplica a los apartados posteriores.

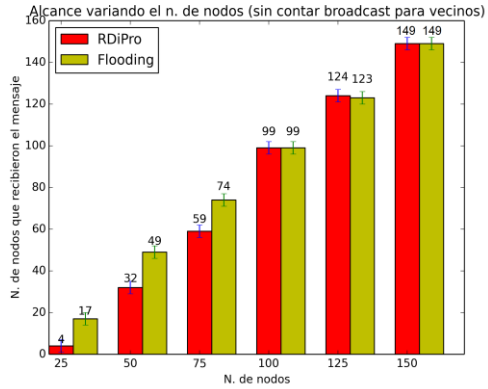

Fig. 2. Resultados del envío de un mensaje con distinta concentración de nodos en la red

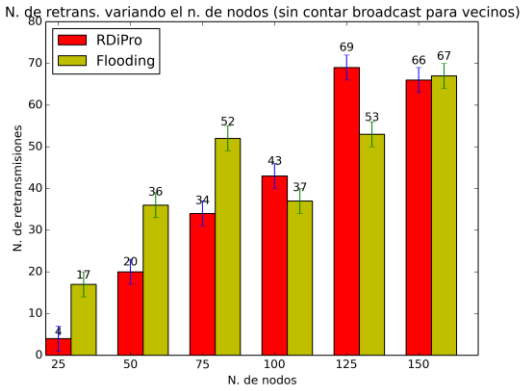

Fig. 3. Resultados del número de retransmisiones totales según la concentración de nodos de la red

B. Variando el número de mensajes de la red

Esta segunda prueba viene a paliar los errores cometidos a visualizar los resultados del envío de un solo mensaje, los que acabamos de ver. Para ello se realiza el estudio enviando mayor número de mensajes para encontrar un valor medio más fiable, pero dado la carga que esto supone para la simulación, sólo se ha realizado estas pruebas para los escenarios de 25 y 50 nodos, variando el número de nodos que envían mensajes desde 5 a 20 nodos, con un paso de 5. En las figuras 4 y 5, vemos el número de nodos que recibieron al menos un mensaje, donde se puede apreciar que todos los nodos lo hicieron en prácticamente todos los casos.

C. Variando el tamaño del mensaje

Concluyendo con este apartado, es importante saber cómo funciona este algoritmo con paquetes de distintos tamaños, el cual si es muy grande tendrá que ser dividido y enviado por el mismo nodo en varias transmisiones. Así, se ha realizado una prueba, cuyos resultados se plasman en las figuras 6 y 7, donde se varía el tamaño de los mensajes. Para esta simulación se han tenido en cuenta pocos nodos por la necesidad de realizar una prueba rápida para ver un primer resultado del funcionamiento. Concretamente, se han creado mensajes de 5 a 20 bytes, con un paso de 5 bytes, y como en las pruebas anteriores, los datos a contabilizar son el número de nodos que han recibido la transmisión, y el número de retransmisiones totales realizadas en toda la red.

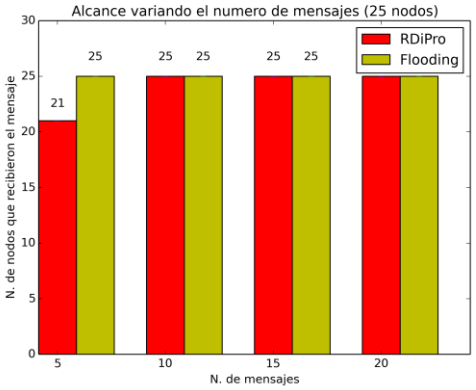

Fig. 4. Resultados del número de nodos que recibieron un mensaje en una red de 25 nodos

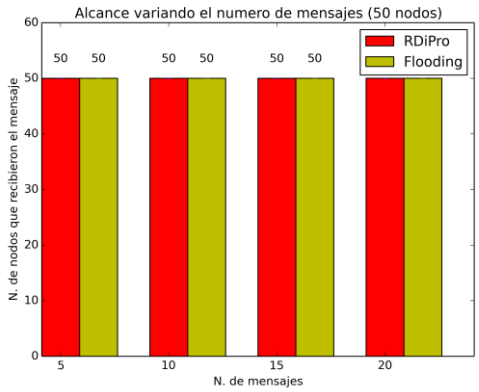

Fig. 5. Resultados del número de nodos que recibieron un mensaje en una red de 50 nodos

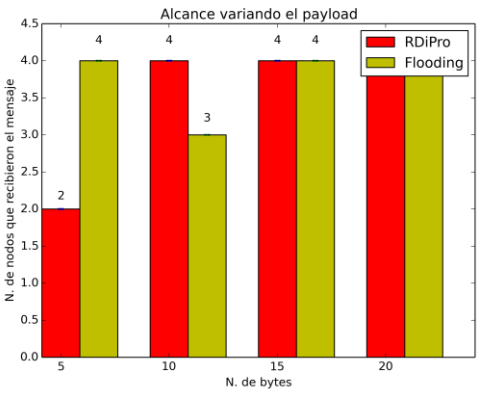

Fig. 6. Resultados del número de nodos que recibieron el mensaje

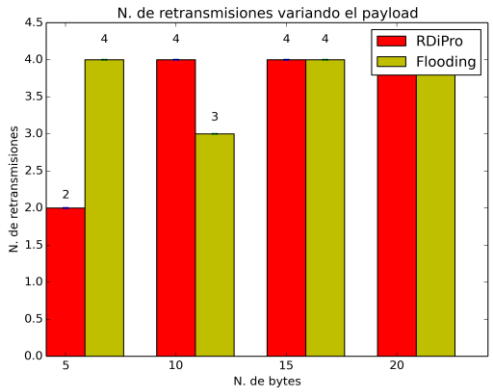

Fig. 7. Resultados del número de retransmisiones totales según el tamaño del mensaje

En los resultados que se muestran se aprecia cómo, salvo casos específicos, no hay diferencia entre estos protocolos bajo el análisis de estos datos.

V. CONCLUSIONES

En este documento hemos presentado RDiPro, un protocolo para diseminación de mensajes en redes de sensores. Hablamos de algunos protocolos que empleaban técnicas parecidas a las que usa RDiPro pero que terminaban fallando en algún aspecto. Con RDiPro se ha pretendido reforzar esas partes en las que flojeaban y además se ha innovado en la forma de hacerlo. El algoritmo aprovecha parámetros que están a disposición de todos los nodos como es la potencia de transmisión base y el RSSI de un mensaje para obtener una distancia relativa a partir de la cual calculará la probabilidad de retransmisión en función de unos umbrales. Permite una gran libertad a la hora de implementar el sistema encargado de obtener la tabla de vecinos siempre y cuando se proporcionen los datos necesarios para el correcto funcionamiento del protocolo. Esto lo hace ideal para adaptarse a posibles cambios de la red. Como futuras mejoras de RDiPro se propone:

- Hacer más robusta la forma de obtener la distancia entre los nodos
- Incluir una mayor cantidad de parámetros para determinar la probabilidad final (como la batería disponible, velocidad de datos, etc.)
- Hacer que los mensajes lleguen a una mayor cantidad de nodos, pues vemos que, aunque el número de retransmisiones disminuye, es posible que el mensaje no llegue a todos

REFERENCIAS

[1] V. Drabkin, R. Friedman, G: Kliot and M. Segal, “RAPID: Reliable Probabilistic Dissemination in Wireless Ad-Hoc Networks,” Comp. Sci. Dept., Technion – Israel Institute of Technology, Israel, 2008.

[2] J. Seo, M. Kim, H. Choo and M. Mutk, “EDAS: Energy and Distance Aware Protocol Based on SPIN for Wireless Sensor Networks,” Comp. Sci. Dept., Sungkyunkwan University, Korea.

[3] P. De, Y. Liu and S. Das, “ReMo : An Energy Efficient Reprogramming Protocol for Mobile Sensor Networks,” Comp. Sci. Dept., Univ. Texas, Arlington, 21 Mar., 2008.