Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Modelado y simulación de redes de sensores inalámbricas subacuáticas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Agustín Roca González

Tutor: Juan Vicente Capella Hernández

Curso 2014/15

Resumen

El presente TFG implementa un protocolo de enrutamiento para redes inalámbricas de sensores subacuáticos. Dicho protocolo, llamado CARP, *Channel-aware Routing Protocol*, en castellano, protocolo de enrutamiento de canal consciente, hace uso de información acerca de la calidad del enlace. Los nodos son seleccionados como transmisores si tienen un historial reciente positivo de retransmisiones a sus vecinos. CARP combina la calidad del enlace con una topología de información simple basada en saltos, siendo capaz de crear una ruta alrededor de vacíos y zonas de sombra, mejorando así sus prestaciones. Estas redes inalámbricas de sensores subacuáticas tienen gran interés por el hecho de que permite recolectar información en zonas de difícil acceso para la tecnología que existía anteriormente ya que no permitían cubrir un área muy grande y tendían a tener un coste elevado. En este trabajo se ha comparado el rendimiento de CARP a través de simulaciones en el ns-3 con el rendimiento de otro protocolo de enrutamiento subacuático llamado DBR. Los parámetros más interesantes incluyen el porcentaje de paquetes perdidos, el retraso de los paquetes entre extremos y el consumo de energía. Los resultados expuestos en este TFG demuestran que CARP es un protocolo más robusto, permitiendo una eficiencia mejor que el protocolo DBR. Asimismo lo supera con creces en cuanto a latencia y consumo de energía.

Palabras clave: redes subacuáticas, capa MAC, CARP, redes inalámbricas de sensores, protocolo de enrutamiento, ns-3.

Abstract

Present TFG implements a routing protocol for underwater wireless sensor networks. This protocol called CARP, Channel-aware Routing Protocol, makes use of information regarding the quality of the link. The nodes are selected as transmitters if they exhibit recent history of successful transmissions to their neighbors. CARP combines the quality of the link with a simple information topology based on jumps, being able to create a route around voids and shadow zones, improving in this way its services. These underwater wireless sensor networks are very interesting because they allow to collect information in areas of difficult access to the previously technology, since they did not allow to cover a very large area and tended to be very expensive. In this work, CARP performance through simulations in ns-3 and another underwater routing protocol performance called DBR have been compared. The most interesting parameters include percentage of lost packets, delay of packets between ends and energy consumption. The showed results in this TFG demonstrate that CARP is a more robust protocol, allowing a better efficiency than the DBR protocol. Also it is way over it as far as delays and energy consumption are concerned.

Keywords: underwater networks, MAC layer, CARP, wireless sensor networks, routing protocol, ns-3.

Tabla de contenidos

1. Introducción

En este trabajo procederemos a estudiar un algoritmo de enrutamiento para redes de sensores subacuáticos. Estas se han postulado en la actualidad como una tecnología en auge, ya que sus aplicaciones están creando un gran impacto por la adquisición de información muy valiosa en diversas áreas de investigación [9][10].

Los principales inconvenientes para el despliegue de las mencionadas redes es que operan en un entorno muy específico y deben adaptarse a las peculiaridades de este medio. Algunas de estas singularidades son: los largos retrasos de propagación, la variabilidad de la velocidad del sonido, la atenuación de la señal y muchos otros impedimentos ambientales. Estas características dificultan la creación de redes subacuáticas. En concreto, la MAC y los protocolos de enrutamiento se enfrentan a nuevos desafíos respecto a sus semejantes terrestres. Por ende, la mejor opción ante este reto es el uso de ondas acústicas, ya que las ondas de radiofrecuencia sufren una gran atenuación en el agua, por lo que el radio de acción se reduciría.

A causa de los grandes avances en las comunicaciones de dichas redes y las mejoras de las infraestructuras, esta tecnología puede ser utilizada en un sinfín de aplicaciones, como por ejemplo: monitorización ambiental, localización y rastreo, predicción de eventos sísmicos submarinos, exploración científica, explotación comercial, movimientos de tierra, control de la polución e información sobre placas tectónicas. Las redes subacuáticas permiten también el estudio del ecosistema marino y la fauna que lo habita. Muchas de estas aplicaciones actualmente están basadas en cableado subacuático lo cual suele ser muy costoso y estar severamente limitado, por lo que no puede ser escalable. Sin embargo, el uso de las comunicaciones mediante sensores inalámbricos permite crear una red más flexible y que permite cubrir un área mayor de exploración.

Estas redes están formadas por un número variable de nodos, dispuestos bajo el agua y en la superficie, cuya función consiste en la realización de tareas sobre un área en concreto. Para conseguir este objetivo, los nodos deben intercambiar y compartir información entre ellos mismos y las estaciones y, simultáneamente, modificar los

canales de comunicación para adaptarse al flujo de información que circula por la red. Figura 1.

Puesto que se trata de redes en un entorno con un despliegue costoso y difícil, los dispositivos reales son caros, por lo que se necesita un medio de simulación que permita ejecutar los algoritmos de una manera lo más realista posible. Estas simulaciones permitirán obtener los parámetros adecuados del protocolo sin el coste económico que suponen los dispositivos reales.

En este caso, hacemos uso de un simulador de redes de eventos discretos llamado ns-3, heredero del simulador ns-2, para la implementación del protocolo. En esta versión, ya están implementadas las capas MAC y física de la pila de protocolos para redes subacuáticas, en las que se incluyen distintos modelos. Asimismo, es conveniente tener en consideración más elementos del simulador que facilitarán la tarea al proporcionar un sistema de posicionamiento de los nodos, simular su movilidad, pudiendo estar estáticos o no y permitiendo determinar su consumo energético.

A este respecto, nuestra tarea será la de implementar un protocolo para sensores inalámbricos. En concreto, el protocolo CARP, por las siglas en inglés *Channel-Aware Routing Protocol*, utiliza la calidad del enlace entre nodos para efectuar la transmisión de la información. Los receptores serán seleccionados solo si tienen un historial de transmisiones satisfactorio a sus vecinos. Además del citado historial, CARP también hace uso de una topología simple basada en saltos para crear rutas alrededor de vacíos y zonas de sombras. Entre varias posibles retransmisiones, un nodo CARP selecciona el vecino con la mayor energía residual y el que mejor calidad de enlace tiene respecto a sus vecinos, permitiendo a los nodos seleccionar los enlaces más robustos y fiables [3][4].

Este protocolo asume que los datos fluyen de manera unidireccional, procedentes de nodos con sensores hasta los sumideros que estarán, generalmente, en la superficie. Los nodos saben a qué distancia se hallan del sumidero, pero no mediante medidas físicas, si no gracias a los saltos que le separan de éste. De tal modo que la información fluirá desde los nodos más alejados del sumidero hasta los más cercanos a éste.

Finalmente como se explicó con anterioridad en el resumen, procederemos al final de nuestro trabajo a la comparación del rendimiento de CARP con el de DBR (*Depth-Based Routing Protocol*), el cual sirvió de base para muchos otros protocolos cuyas rutas se basan en la profundidad de los nodos.

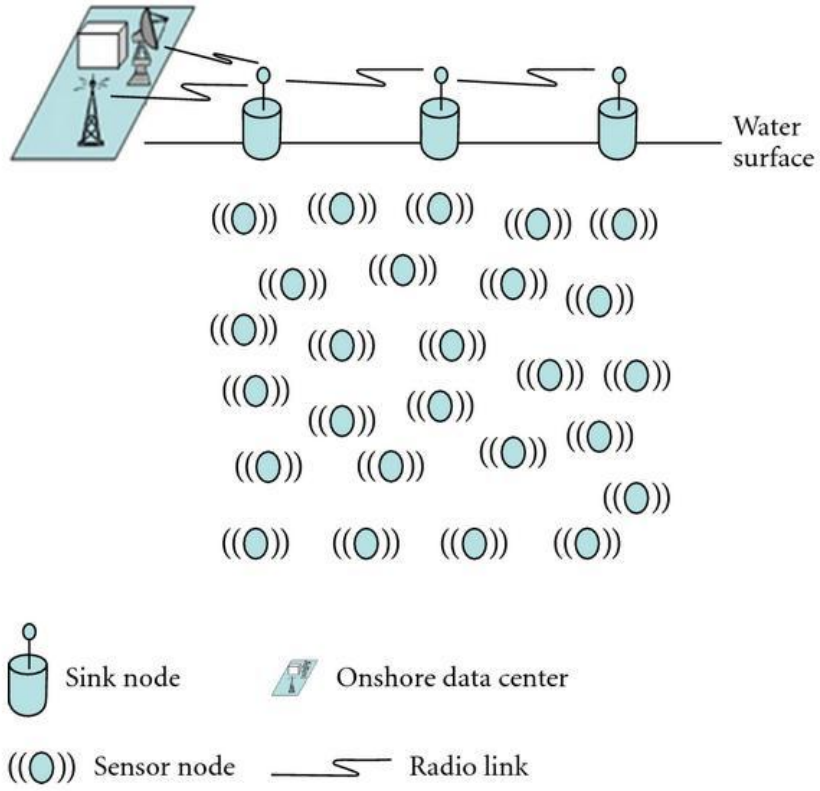

Figura 1. Esquema general de una red de sensores subacuática.

1.1 Objetivos del proyecto

Este proyecto tiene como primer objetivo implementar el protocolo de enrutamiento CARP en el simulador ns-3. Tras ser implementado, lo estudiaremos y simularemos intensivamente y lo compararemos con el protocolo DBR.

Esto nos permitirá adquirir conocimientos y estudiar en profundidad el módulo para redes subacuáticas del simulador ns-3. La capa física implementada nos otorga el canal, los modelos de propagación, SNR… y la MAC distintas propuestas entre las que elegir. De esta forma, nuestro protocolo añadirá una nueva capa aún no explorada en la suite.

1.2 Estructura de la memoria

En el capítulo 1 se ofrece una introducción a las redes subacuáticas de sensores. Haremos hincapié en la implementación del protocolo CARP. Tras ello, en el capítulo 2, se presentará el trabajo relacionado, donde resumiremos varios tipos de protocolos de enrutamiento para entornos subacuáticos. A continuación, se describirán con detenimiento las herramientas utilizadas para el desarrollo del proyecto. En el capítulo 4 se explicará la implementación del protocolo a desarrollar, en este nuestro caso, el protocolo de enrutamiento CARP y donde se describirán y explicarán las decisiones tomadas para su implementación. Después, en el capítulo 5, se detallarán los experimentos realizados y se mostrarán y analizarán los resultados obtenidos. Finalmente se presentarán las principales conclusiones obtenidas durante la realización del proyecto así como las futuras líneas de trabajo abiertas.

2. Trabajo relacionado

Los protocolos para comunicaciones subacuáticas han recibido recientemente más atención. En la actualidad, se conocen tres familias de protocolos de enrutamiento: proactivos, activos y geográficos.

Los proactivos (ej. DSDV) realizan un gran trabajo a la hora de establecer las rutas por primera vez o cada vez que la topología de la red cambia debido a los movimientos de los nodos o los fallos de éstos, ya que dicho cambio de topología debe ser transmitido a todos los nodos. Por lo que cada dispositivo puede establecer un camino a cualquier nodo de una red [1].

Los reactivos (ej. AODV) son apropiados para entornos más dinámicos. Cada nodo guarda información solo de las rutas activas y éstas son determinadas cuando van a ser datos enviados [2].

Los geográficos (ej. PTKF) tienden a soportar mejor la escalabilidad. Sin embargo, los receptores GPS no funcionan correctamente en el entorno subacuático y deben conocer las tres dimensiones del nodo para poder reenviar los paquetes, cosa que es un problema nada trivial de resolver [5].

Entre los primeros protocolos que abordaron el problema de encontrar rutas encontramos el VBF (*Vector-Based Forwarding*). Los nodos redirigen los paquetes mediante *broadcast* a otros nodos que tiene almacenados en un vector con un radio predeterminado. La eficiencia de este vector reside principalmente en el radio de acción, ya que si es muy grande muchos nodos recibirán el paquete y por ende provocará muchas retransmisiones, lo que a su vez conlleva interferencias y duplicados. En cambio si es muy pequeño puede que no encuentre a ningún nodo en dirección al sumidero.

Debido a que el uso de información geográfica para entornos submarinos es problemático de tratar, existen ciertos protocolos que emplean información parcial, como es el caso de la profundidad, dado que puede ser fácilmente determinada con gran precisión. Este es el caso del DBR (*Depth-Based Routing protocol*), donde cada nodo que recibe un paquete de información lo reenvía únicamente si su profundidad es menor respecto al nodo emisor del paquete (y además el paquete no ha sido ya reenviado con anterioridad). Para ello, el nodo espera un tiempo prudencial antes de enviarlo, basado en la diferencia de profundidades entre el nodo emisor y el receptor, por lo que cuanto más grande sea la diferencia, menor tiempo de espera tendrá. Así pues, poseerán más prioridad de envío cuanto más cerca del sumidero se encuentren. Si en dicha espera el paquete ya fue retransmitido simplemente lo desecha [7] [8].

3. Herramientas

Para la realización del proyecto utilizamos el simulador ns-3, concretamente la versión 3.10. Se trata de un simulador de eventos discretos, ampliamente conocido por la comunidad académica e investigadora. Permite simular tanto protocolos *unicast* como *multicast* y es muy empleado en la investigación de redes ad-hoc. Implementa protocolos tanto de redes cableadas como de redes inalámbricas.

Está escrito en C++ y cuenta con una gran diversidad de módulos que permiten modelar diversos aspectos interesantes a estudiar: su consumo energético, la distribución espacial de los nodos, la movilidad de éstos y por supuesto, el de mayor relevancia para nosotros: la capa de red física y MAC para redes subacuáticas.

La versión actual es un *fork* de la antigua versión ns-2, que estaba escrita en C y hacía uso de un lenguaje de scripting basado en Tcl para definir los test. En 2005 se empezó a desarrollar la nueva versión ns-3 desde cero y basada exclusivamente en C++, en la que se aprovechó para solventar algunas de las deficiencias de la versión anterior, especialmente las referidas al lenguaje de scripting y la falta de modularización. En 2008 se lanzó la primera versión de esta rama [6].

4. Implementación

4.1 Descripción del protocolo *CARP (Channel-aware Routing Protocol)*

4.1.1 Inicialización de la red y configuración del contador de salto

Channel-aware Routing Protocol es un protocolo de enrutamiento especialmente diseñado para las redes de sensores subacuáticas. El protocolo, para que funcione de manera óptima, debe de iniciar la red antes de empezar a transmitir paquetes de información. Para poder iniciarla y que cada nodo sepa a qué distancia del sumidero esta, los paquetes *HELLO* fluyen desde el sumidero hasta que llegan a todos los nodos de la red. De este modo, cada nodo recibe su contador de salto HC(*x*), que es su distancia, en saltos, al sumidero. Cada paquete *HELLO* transporta información de su nodo emisor src (inicialmente, será el ID del sumidero) y el valor del contador de salto: *HELLO* = <src,HC>. El sumidero genera el primer paquete *HELLO*, configurando su propio contador de salto a cero, y transmitiéndolo mediante *broadcast* a todos sus vecinos a distancia uno de él. Cada nodo que recibe un paquete *HELLO* comprueba si su propio contador de salto es mayor que el contador de salto que transporta el paquete más uno. Si es el caso, el nodo actualiza su propio contador de salto al valor que transporta el paquete *HELLO* más uno y retransmite el paquete incrementando en uno el campo del contador. En cualquier otro caso, descarta el paquete ya que su contador sería mejor que el recibido por el paquete *HELLO*. Al final del proceso de inundación de paquetes *HELLO* cada nodo sabe con precisión su distancia en saltos al sumidero e

información sobre sus vecinos más cercanos al sumidero. Todo esto queda esquematizado en la figura 2.

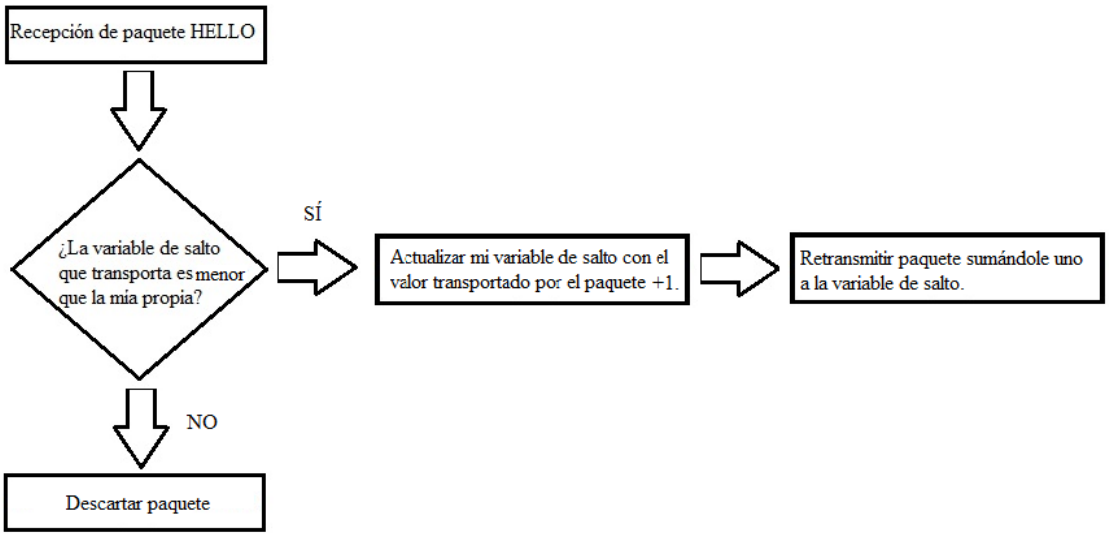

Figura 2. Esquema que muestra que se hace tras recibir un paquete *HELLO*.

4.1.2 Reenvío de información

Cuando un nodo *x* tiene uno o más paquetes de información que reenviar elige el receptor más adecuado entre todos sus vecinos. La búsqueda de dicho nodo es inicializado por el nodo *x* mandando mediante *broadcast* un paquete de control llamado PING, el cual transporta el identificador único del nodo *x*, el contador de salto de *x*. Figura 3.

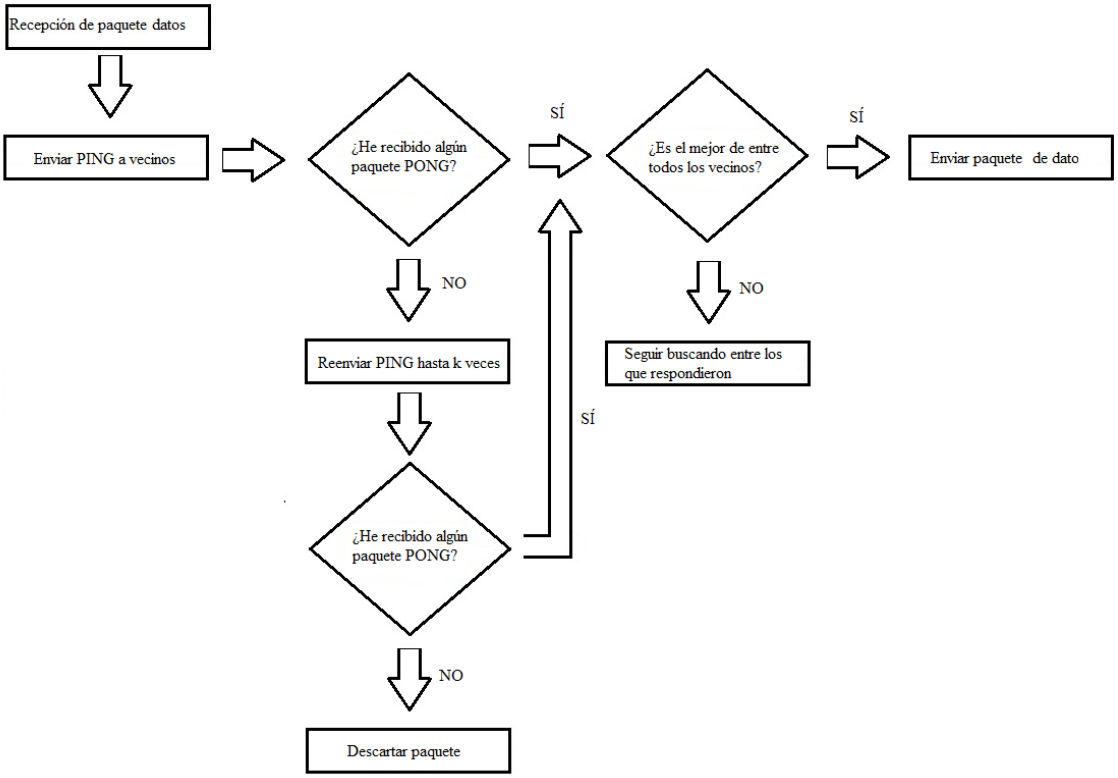

Figura 3. Esquema que muestra que se hace tras recibir un paquete de datos.

Un nodo *y* que reciba el paquete *PING* responderá inmediatamente con un paquete PONG al nodo *x* que transmitió el paquete PING siempre y cuando HC(*y*)≤HC(*x*). En caso contrario, el nodo *y* lo descartará. El paquete PONG contiene los identificadores del nodo *x* e *y*, el contador de salto de *y*, su contador de salto propio HC(*y*), la energía residual del nodo *y* y finalmente lq, un indicador de la calidad del enlace del nodo *y*. Figura 4.

La elección del nodo que reenviará los datos sucede de esta forma. Tras enviar un paquete PING, el nodo *x* espera la recepción de paquetes PONG durante un tiempo δ. El tiempo de espera δ se establece para un tiempo de ida y vuelta más el tiempo que tarda en enviarse un paquete PING y PONG al vecino más lejano, así pues este tiempo variará según las condiciones marítimas. Si el nodo *x* no recibe ningún paquete PONG de sus vecinos continúa intentándolo hasta *k* veces. Si tras todos estos intentos, sigue sin recibir ninguna respuesta descarta el paquete. Tras esperar un tiempo δ, el nodo *x* utiliza el valor que indica la calidad del enlace *y* transportado en el paquete PONG.

El nodo y con mejor ratio lq_y/HC(y) será elegido como receptor y le serán transmitidos todos los paquetes de información. Mediante esta fórmula, los nodos más cercanos al sumidero tendrán más preferencia y los más alejados solo serán escogidos si su calidad de enlace es significativamente mejor que el resto.

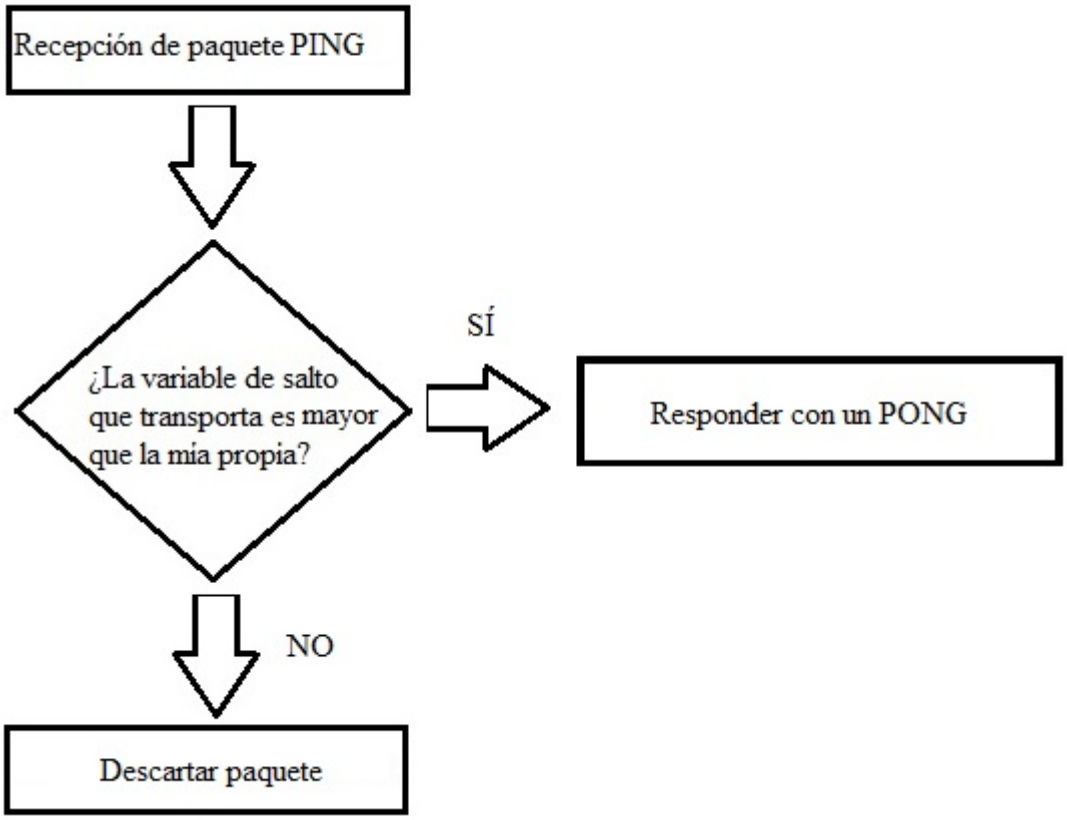

Figura 4. Esquema que muestra que se hace tras recibir un paquete *PING*.

4.1.3 Cálculo de la calidad del enlace lq

La calidad $lq_{y,z}$ del enlace de un nodo y se calcula en base a las transmisiones con éxito a sus vecinos z. Es fundamental mantenerla actualizada ya que tiene un gran peso a la hora de que un nodo x busqué vecinos para retransmitir datos. Está definida de manera exponencial, donde las transmisiones recientes tienen más peso que las más antiguas a la hora de establecer la calidad del enlace por lo que permite al protocolo tener en cuenta la naturaleza variable del canal.

Por el canal circulan dos tipos de transmisiones: los paquetes PING/PONG y la transmisión de paquetes de datos. Para cada transmisión τ del nodo *y* al nodo vecino *z* el primero computa:

$$lq^{\tau}{}_{y,z} = \alpha Y^{\tau}{}_{y,z} + (1-\alpha)\, lq^{\tau-1}{}_{y,z}$$

El coeficiente $\alpha \in (0,1)$ es la constante que permite controlar como de rápido decrece el valor de las transmisiones más antiguas. Por ejemplo, un alto valor de α podría ser utilizado para canales muy variables. El término $Y^{\tau}{}_{y,z}$ es el porcentaje de τ retransmisiones acertadas desde *y* hasta *z*. El término $lq^{\tau-1}{}_{y,z}$ es la media tras τ-1 transmisiones desde *y* hasta *z*. La ecuación (1) es recursiva, definiendo $lq^{\tau}{}_{y,z}$ como el porcentaje de éxito de la primera transmisión. El valor lq que el nodo *y* transmite en su paquete PONG al nodo *x* es el mejor de entre todos los $lq^{\tau}{}_{y,z}$ a sus vecinos *z* que pueden encontrar un retransmisor en dirección al sumidero.

4.2 Desarrollo del protocolo

Para el desarrollo del protocolo se han implementado dos clases principales. Por un lado la clase CARPHeader() y por otro la clase CARPNode(). La figura 5 muestra los elementos más importantes de estas clases.

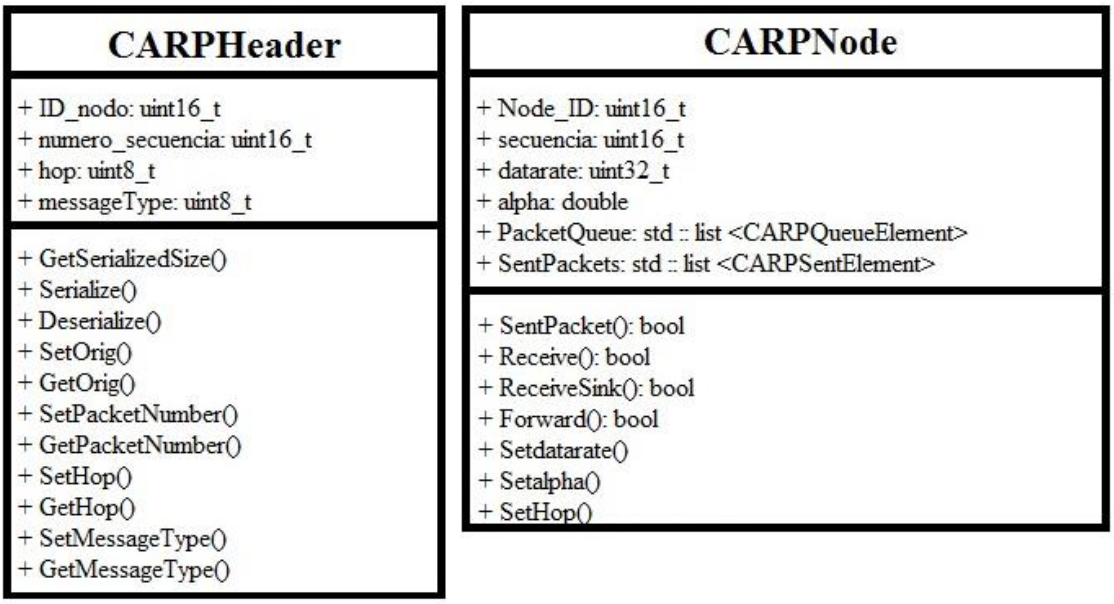

Figura 5. Diagrama con los elementos principales de ambas clases.

La clase CARPHeader contiene los métodos necesarios para crear las cabeceras que se agregarán a los paquetes. A cada paquete enviado le añadiremos dicha cabecera para que pueda ser manejada por el protocolo y que éste pueda retransmitir los datos correctamente. Esta clase se encargará de serializar y deserializar los campos de la cabecera. Para la codificación de estos campos se utilizará un tipo uint16_t para el ID del nodo origen, un tipo uint16_t para el número de secuencia, un tipo uint8_t para indicar la distancia hasta el sumidero y un tipo uint8_t para indicar el tipo de mensaje. Todo ello suma un tamaño total de la cabecera de 6 bytes. Podríamos utilizar un tipo de menor tamaño para el número de secuencia, dependiendo de la cantidad de paquetes que vayan a ser enviados por unidad de tiempo. No se utiliza un tipo uint8_t para la dirección de origen de los paquetes (que es el tamaño con que se almacena dentro de la clase UanAddress ya implementada en el ns-3) pues limita el número de nodos a 255, lo cual introduciría acotaciones a la hora de realizar simulaciones con un número de nodos superiores.

Por otro lado se ha creado la clase CARPNode que instalaremos en los nodos que deban tratar con este protocolo. Tiene dos funciones principales: Receive() y Send().

Receive() se encargará de recibir los paquetes que lleguen hasta el nodo, y será invocado por las capas inferiores de la pila de protocolos. Tomará una primera decisión sobre descartarlo o prepararlo para ser retransmitido.

Send() se encargará de llamar a las capas inferiores con el paquete que haya de ser enviado, ya con su correspondiente cabecera insertada correctamente.

Otro método de gran importancia es Forward(), que se encargará de recibir los paquetes aceptados para el reenvío por Receive().

4.3 Descripción de la cabecera

La cabecera añadida a cada paquete tiene una valiosa importancia para el correcto funcionamiento del protocolo. La clase CARPHeader ha sido implementada y consta de cuatro campos:

- Hop: Es la distancia, en saltos, del nodo al sumidero. Sirve para encontrar rutas hacia el sumidero de la manera más óptima. A la hora de escoger nodos que retransmitan paquetes de datos, aquellos con menor distancia serán más comúnmente elegidos para encaminar la información hacia la superficie. Utilizamos un entero sin signo de 8 bits, ya que con el número de nodos tratados y la media de distancia entre los extremos y el sumidero abarcamos ampliamente todas las posibilidades. Inicialmente el valor está puesto a 255, una vez reciba el paquete HELLO adquirirá el valor que dicho paquete transmita.

- NodeID: Es el identificador del nodo, el cual permite referirlo de manera única, que se almacena como un entero sin signo de 16 bits. No se hace uso de un almacenamiento de 8 bits pues éste reduciría el número posible de nodos a 255, impidiéndonos realizar simulaciones con un número mayor de nodos. Por ello empleamos 16 bits, lo que nos proporciona más de 65000 posibles nodos, que apreciamos es suficiente volumen de nodos para realizar nuestras simulaciones. Al instalar el nodo, se le asigna un valor.

- Sequence number: Es el número de secuencia que permite identificar de manera única a cada paquete perteneciente a un mismo nodo. Utilizamos un entero sin signo de 16 bits. Dependiendo del número de envíos por unidad de tiempo, podría reducirse a 8 bits, pero para evitar posibles complicaciones elegimos uno de 16. Se inicializa a 0 al arrancar y se incrementa en uno con cada paquete enviado.

- MessageType: Es el campo que indica el tipo de mensaje que hay en el paquete. Lo hemos codificado como un entero sin signo de 8 bits para tratar todos los tipos de mensajes diferentes.

5. Experimentación

5.1 Distribución espacial

Los nodos han sido distribuidos uniforme y estáticamente y de manera aleatoria en un espacio tridimensional con forma de cubo de 500m x 500m a distintas profundidades. En la superficie está colocado el nodo sumidero, a una profundidad de 0 metros. La coordenada Z indica la profundidad y se expresa como un valor negativo. De esta forma Z=0 sería la superficie, -1 correspondería a un metro de profundidad, etc…

Todos los nodos, excepto el sumidero, se han distribuido mediante el RandomBoxPositionAllocator. Como ya hemos dicho, el nodo sumidero se encuentra en el centro de la superficie.

Para acabar, a cada nodo se le instaló una batería y dependiendo de si era nodo sumidero o no tenía una capacidad u otra, para así poder medir la energía consumida y restante a lo largo de la simulación. En concreto, el nodo sumidero fue configurado con más capacidad de energía que los nodos intermedios. Los consumos de este dispositivo son los siguientes: 1.5W para transmisión de paquetes de control, y 3.3W para transmisión de paquetes de información, 620mW para recepción y 85mW para idle.

5.2 Parámetros utilizados

Para la configuración de la capa física de la red UAN se utiliza el método estándar proporcionado por el simulador para el cálculo del SNR (ratio señal-interferencia), así como el método estándar de cálculo de errores en los paquetes (PER).

Se asume una modulación BPSK, con una frecuencia central de 24kHz para un ancho de banda de 2kHz. La construcción de la topología asegura que cada nodo encontrará al menos una ruta que le lleve al sumidero a través de enlaces lo suficientemente robustos (con respecto al SNR).

5.3 Parámetros a investigar

La efectividad del protocolo será estudiado en base a los siguientes parámetros.

- Porcentaje de paquetes recibidos, definido como la relación entre los paquetes generados por los nodos y los recibidos por el sumidero.

- Latencia de paquetes, definido como el tiempo entre la generación del paquete y la correcta recepción en el sumidero.

- Energía consumida, definido como la energía consumida por los nodos al finalizar la simulación.

5.4 Resultados

Tras la ejecución de diversas simulaciones para los parámetros descritos anteriormente, enviando 300 paquetes, a razón de uno por segundo, un datarate de 2000 bps, obtenemos los siguientes resultados.

Los valores obtenidos los compararemos, como se propuso en el paper, con los obtenidos por el protocolo DBR en situaciones similares. A la hora de realizar simulaciones del protocolo DBR ha sido usado el expuesto según la investigación en [7].

En primer lugar, se estudia la latencia de los paquetes desde el nodo hasta el sumidero. Figura 6.

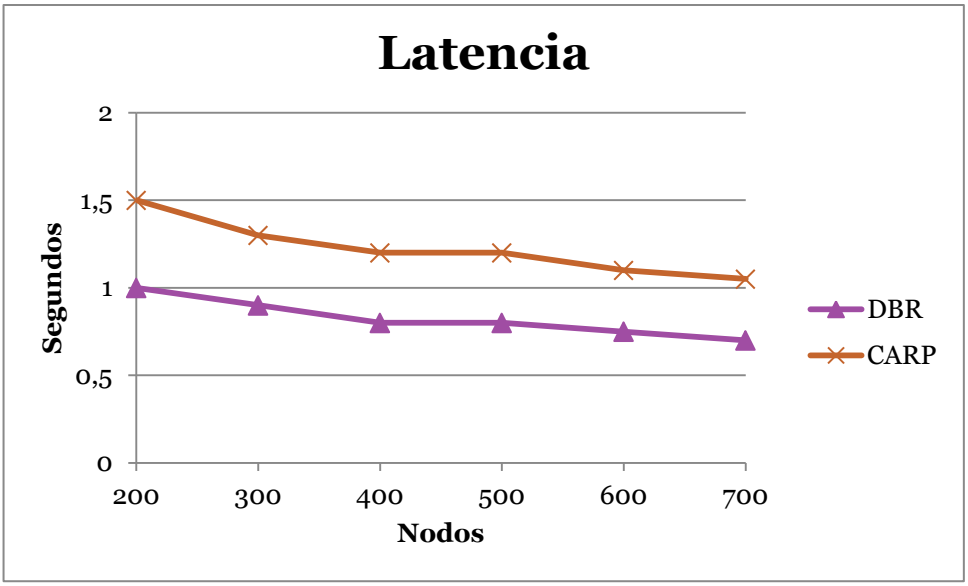

Figura 6. Gráfica que muestra la latencia entre ambos protocolos.

Ambos protocolos muestran un buen rendimiento. Incrementar el número de nodos permitirá reducir el tiempo que tarda un paquete desde que sale del origen hasta que llega al sumidero como es esperado. CARP muestra el mejor rendimiento, debido a la selección del mejor nodo vecino para retransmitir los datos, basado en la calidad del enlace. La razón por la que la latencia de DBR es ligeramente superior es porque tiene rutas más largas a su disposición. Los paquetes de datos generados en el centro de la red avanzan verticalmente, pero los generados en los límites de la red avanzan primero verticalmente y luego, cuando están cerca de la superficie, de manera horizontal hasta los sumideros. Por esta razón las rutas tienden a ser más largas y por consecuencia adquiere mayores retardos. Esto último tendrá una influencia directa sobre el consumo total. En cambio, CARP al hacer uso del mejor vecino, solo retransmitirá los paquetes de datos una vez en dirección al sumidero y siempre hacia arriba, restando en uno la diferencia en saltos hasta éste. Al no ser una transmisión *broadcast* se evitan las colisiones que se traducen en retardos más largos.

En segundo lugar, se analiza el porcentaje de paquetes recibidos en el sumidero de manera correcta respecto al número de paquetes generados por los nodos. Figura 7.

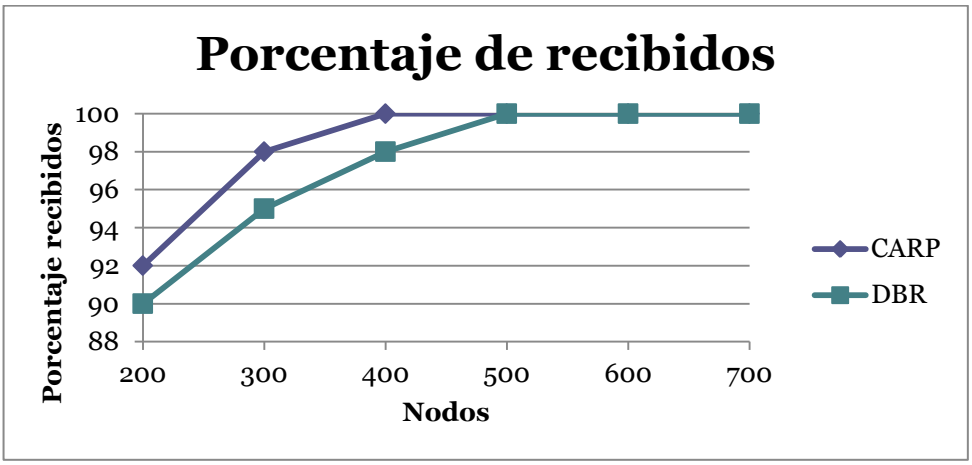

Figura 7. Gráfica que muestra el rendimiento entre ambos protocolos.

Como puede observarse en la figura 7, cuando el número de nodos aumenta, la probabilidad de que el paquete sea recibido correctamente por el sumidero lo hace también. Dicha probabilidad tiende al 100% conforme los nodos van aumentando. Esto es debido a que hay mayores nodos que pueden formar parte de la ruta, por lo que la probabilidad de encontrarte un vecino en dirección al sumidero crece conforme la red lo hace. Aquí no hay tantas diferencias entre CARP y DBR ya que ambos funcionan dirigiendo la información hacia nodos superiores, por lo que en cualquier caso, los paquetes encontrarán más fácilmente rutas hacia los sumideros conforme más nodos haya en la red.

En último lugar, se estudia el consumo de energía total por parte de los nodos tras la simulación. Figura 8.

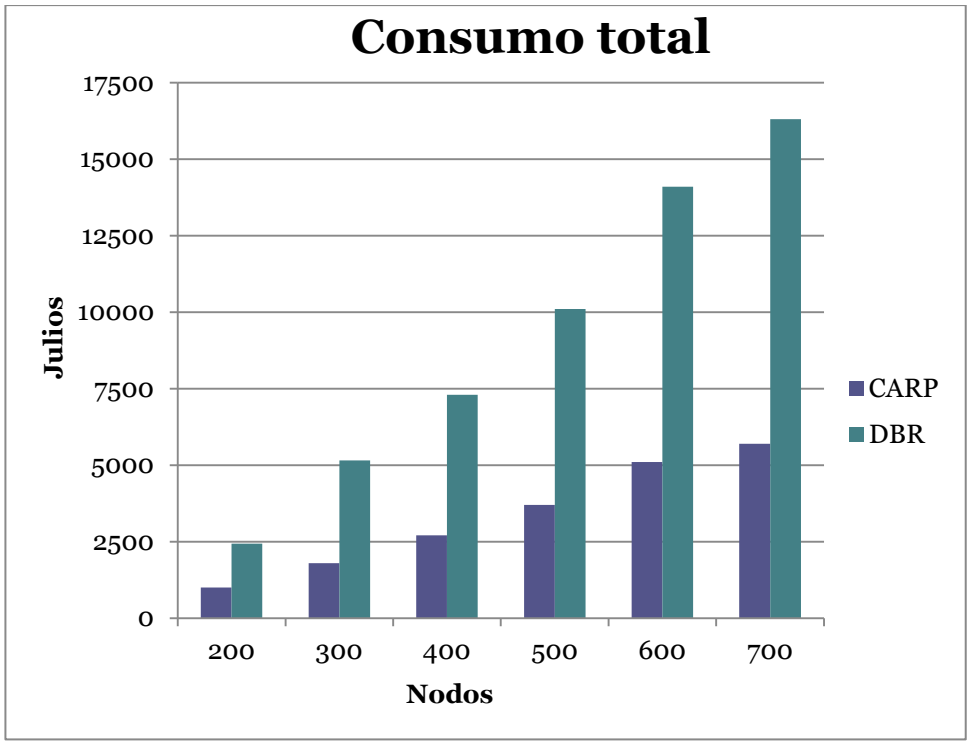

Figura 8. Gráfica que muestra el consumo total entre ambos protocolos.

Se considera que no solo la energía gastada es empleada en la transmisión y recepción de paquetes, sean del tipo que sea y que estar escuchando el canal cuando el tráfico de paquetes es bajo puede llegar a consumir más que la transmisión o recepción de paquetes.

Como se puede observar en la figura 8, el protocolo DBR consume más (sobre tres veces más) que CARP. Esto se debe a que DBR es un protocolo de inundación, por lo que incurre en un mayor número de transmisiones. Además, como ya dijimos hace dos páginas, al final del primer párrafo, al ser DBR un protocolo que exige rutas más largas, su demanda de energía también crece. Por el contrario, CARP aprovecha muy bien los enlaces y evita saturar la red utilizando los nodos necesarios, seleccionados de manera exhaustiva, para así propagar la información.

6. Conclusiones y trabajo futuro

De todo lo dicho hasta ahora en este trabajo puede concluirse que CARP se presenta como un protocolo eficiente para redes de sensores inalámbricos subacuáticos. En particular, aprovecha la calidad del enlace para incrementar la eficiencia del protocolo y crear rutas más estables y robustas. Una comparativa basada en simulaciones evalúa el rendimiento entre CARP y DBR y revela que la inclusión de la calidad del enlace para la generación de rutas es clave para la obtención de mejores resultados tanto en rendimiento, en latencia y en consumos de energía.

En el estudio que hemos realizado sobre CARP se confirman los resultados presentados en el artículo.

Como trabajo futuro se propone el estudio del coeficiente α para que se adapte en tiempo real a las variaciones de las condiciones medioambientales. En nuestro caso está establecido a un valor fijo de 0.7, pero el coeficiente arriba citado tiene una importancia vital, pues tiene repercusión a la hora de elegir el vecino que retransmitirá los paquetes de datos, por lo que debe cambiar conforme lo hagan las condiciones medioambientales.

7. Bibliografía

[1] C. Perkins and P. Bhagwat. Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers. In Proc. of ACM Special Interest Group on Data Communications (SIGCOMM). London, UK, 1994.

[2] Mehran Abolhasan, Tadeusz Wysocki, Eryk Dutkiewicz. A review of routing protocols for mobile ad hoc networks, Elsevier 2013.

[3] Stefano Basagni, Chiara Petrioli, Roberto Petroccia and Daniele Spaccini: Channel-aware Routing for Underwater Wireless Networks, IEEE 2011.

[4] Stefano Basagni, Chiara Petrioli, Roberto Petroccia and Daniele Spaccini: CARP: A Channel-aware Routing Protocol for Underwater Acoustic Wireless Networks. Accepted for publication in Elsevier Ad Hoc Networks and Physical Communication, Special Issue on Advances in Underwater Communications and Networks, 2014.

[5] T. Melodia, D. Pompili, I.F. Akyildiz, Optimal local topology knowledge for energy efficient geographical routing in sensor networks, in: Proceedings of IEEE INFOCOM'04, Hong Kong SAR, PRC, March 2004.

[6] Homepage of ns-3: http://www.nsnam.org/

[7] H. Yan, Z. J. Shi, and J.-H. Cui, DBR: Depth-based routing for underwater sensor networks, in Proceedings of the 7th international IFIP-TC6 NETWORKING 2008, May 5–9 2008, pp. 72–86.

[8] Daniel Mazon Menendez: DBR: Depth-Based Routing Protocol para redes de sensors subacuáticos, Sept 2011.

[9] Muhammad Ayaz, Azween Abdullah, Ibrahima Faye, Yasir Batira: An efficient Dynamic Addressing based routing protocol for Underwater Wireless Sensor Networks, in journal of Computer Communications, Nov 2011.

[10] A. Gkikopouli, G. Nikolakopoulos and S. Manesis: A Survey on Underwater Wireless Sensor Networks and Applications, Barcelona, Spain, July 3-6, 2012.