UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Modelado y simulación de un protocolo de comunicaciones para redes subacuáticas

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Izan Catalán Gallach
Tutor: Juan Vicente Capella Hernández
Tutor Experimental: Jose Navarro Alabarta

Curso 2018-2019

Resum

Internet of Underwater Things (IoUT) és una nova classe de IoT que podria definir-se com una xarxa d'objectes intel·ligents subaquàtics interconnectats, i s'espera que este paradigma possibilite una gran quantitat d'aplicacions, contribuint no sols al desenvolupament d' *smart cities/industry*.

L'IoUT es basa en les *Underwater Wireless Sensor Networks (UWSNs)* , que tenen característiques específiques respecte les tradicionals WSN, a l'utilitzar senyals acústics en compte de radiofreqüència, com el retard de propagació, limitat amplada de banda i baixa fiabilitat, a més de resultar especialment crític l'eficiència energètica, representant tot això grans reptes.

En este TFG es modelarà i avaluarà l'adaptació d'un protocol de comunicacions de l'equip d'investigació per a contribuir a possibilitar l'IoUT considerant els reptes plantejats. Per això s'implementarà i adaptarà la base del protocol DBR (*Depth-Based Routing*) , validant-ho i comparant-ho per a entendre el seu funcionament i els paràmetres amb què funciona, realitzant proves sobre cada un d'ells amb la fi de detallar i explicar els resultats del protocol en cadascuna de les seues simulacions.

Una vegada comprés açò s'evolucionarà a un protocol desenvolupant Clustering sobre la base del DBR per a veure els avantatges que presenta respecte a l'anterior i si val la pena la seua utilització, tant en prestacions de consum com de paquets rebuts. Per això i a la fi, s'efectuaran una comparació total.

Paraules clau: Internet of Underwater Things (IoUT), IoT, Xarxes Subaquàtiques (UWSNs), modelatge, protocols d'encaminament"

Resumen

El *Internet of Underwater Things (IoUT)* es una nueva clase de IoT que podría definirse como una red de objetos inteligentes subacuáticos interconectados, y se espera que este paradigma posibilite un gran abanico de aplicaciones, contribuyendo no solo al desarrollo de *smart cities/industry*.

El IoUT se basa en las *Underwater Wireless Sensor Networks (UWSNs)*, que tienen características específicas respecto las tradicionales WSN, al utilizar señales acústicas en lugar de radiofrecuencia, como el retardo de propagación, limitado ancho de banda y baja fiabilidad, además de resultar especialmente crítico la eficiencia energética, representando todo ello grandes retos.

En este TFG se modelará y evaluará la adaptación de un protocolo de comunicaciones del equipo de investigación para contribuir a posibilitar el IoUT considerando los retos planteados. Para ellos se implementará y adaptará la base del protocolo DBR (*Depth-Based Routing*), validándolo y comparándolo para entender su funcionamiento y los parámetros con los que funciona, realizando pruebas sobre cada uno de ellos con el fin de detallar y explicar los resultados del protocolo en cada una de sus simulaciones.

Una vez comprendido esto se evolucionará a un protocolo desarrollando Clustering sobre la base del DBR para ver las ventajas que presenta respecto al anterior y si merece la pena su utilización, tanto en prestaciones de consumo como de paquetes recibidos. Para ello y al final se efectuará una comparación total.

Palabras clave: Internet of Underwater Things (IoUT), IoT, Redes subacuáticas (UWSNs), modelado, protocolos enrutamiento

Abstract

The Internet of Underwater Things (IoUT) is a new class of IoT that could be defined as a network of interconnected underwater intelligent objects, and this paradigm is expected to enable a wide range of applications, contributing not only to the development of Smart cities/industry.

The IoUT is based on the Underwater Wireless Sensor Networks (UWSNs), which have specific characteristics regarding traditional WSN, using acoustic signals instead of radio frequency, such as the propagation delay, limited bandwidth and low Reliability, besides being especially critical the energy efficiency, representing all this great challenges.

This GFR will model and evaluate the adaptation of a communications protocol of the research team to contribute to enable the IoUT considering the challenges planned.

For them, the DBR (Depth-Based Routing) protocol's base will be implemented and adapted, validating and comparing it to understand its operation and the parameters with which it works, performing tests on each of them in order to detail and explain the results of the protocol in each of its simulations.

Once this is understood, we will evolve to a protocol developing Clustering on the basis of the DBR to see the advantages that it presents with respect to the previous one and if it is worth its use, both in consumer benefits and in packages received. For this and in the end a total comparison will be made.

Key words: Internet of Underwater Things (IoUT), IoT, Underwater networks (UWSNs), modeling, routing protocols

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Las investigaciones sobre las redes de comunicaciones inalámbricas durante los últimos años han sido múltiples. No obstante, es habitual que las comunicaciones subacuáticas, siendo éstas también inalámbricas, estén siendo un área en emergente investigación por características tales como el medio.

En los últimos años, las redes inalámbricas constan de nodos (normalmente de no muy alto coste y consumo) dotados de sensores, en un área determinada ya que conjuntamente recogen información del medio comunicándose con un nodo sumidero o puerta de enlace que posteriormente la trata. Estos nodos pueden obtener su energía de baterías, por ello es limitada, con lo que es crítico desarrollar protocolos de comunicaciones que optimicen este consumo.

Estas redes de sensores tienen suma importancia a nivel industrial, militar y ya en menor importancia, en entornos domésticos. Sin embargo, este tipo de redes también se está introduciendo en los ambientes subacuáticos en aplicaciones como la monitorización ambiental, el cartografiado de entornos subacuáticos y su exploración así como la asistencia a la navegación o la prevención de catástrofes naturales gracias a la vigilancia y al reconocimiento de actividades submarinas por los sensores[8].

1.1 Motivación

Este proyecto me supone un gran paso en el aprendizaje del grado de ingeniería informática. En él, siempre había pensado en las redes inalámbricas con aplicaciones a casos del día a día, cómo la conexión sin cables WI-FI. Sin embargo al averiguar todas las aplicaciones que este tipo de comunicaciones inalámbricas tienen en el entorno acuático y lo poco divulgado que están me despertó mi interés a profundizar en esta temática.

Sin embargo también me ofrece la oportunidad de modificar y crear a partir de un protocolo, otro diferente, con otras características y posteriormente ver cual de los dos obtiene las mejores prestaciones. Este tipo de investigación por la creación de algo fructífero y mejor a partir de algo ya existente me hizo interesarme aun más y motivarme para ver si era capaz de conseguir ese reto.

1.2 Objetivos

Debido al interés que despierta esta tecnología de redes subacuáticas y que es en la actualidad cuando se están implementando investigaciones para responder a las necesidades y usos de este tipo de redes gracias a su complejidad y al abaratamiento del coste

de los nodos. Es importante investigar un protocolo de encaminamiento que organice el envío de información de manera eficiente debido a que el número de sensores necesarios en la red es alto y un coste alto de cada elemento haría económicamente inviable su implantación.

Además, las redes de sensores subacuáticas resultan una tecnología emergente y con un gran abanico de aplicaciones prácticas, por lo que son un campo ideal para investigar. Ello explica el creciente número de trabajos sobre el tema, pues son muchos todavía los retos que deben superarse, como introducción de mecanismos de tolerancia a fallos y seguridad, mejora de las prestaciones, aumentar la eficiencia energética, soporte a la movilidad, etc.

En esta línea y debido a la gran cantidad de protocolos y configuraciones existentes es importante investigar y encontrar una óptima implementación para evitar problemas en las redes. Por ello, el principal objetivo de este trabajo es comparar dos tipos diferentes de protocolos de encaminamientos en red (uno implementado sobre la base de otro, demostrando así compatibilidad y aprovechamiento de lo ya previamente funcional) siendo como posteriormente se indica uno reactivo y otro clusterizado. Se pretende comprobar si es viable realizar esta implementación y si merece la pena en cuanto a productividad, dificultad o complejidad y mejora. Después, el objetivo es llevar esta mejoría a la solidez y estabilidad, convirtiendo los protocolo de clusters viables.

A parte, el reto también consiste en estudiar, aprender y conocer los protocolos de enrutamiento más estudiados, de ese modo se podrá saber si el resultado ha tenido éxito o no.

Por ello, este Trabajo de Fin de Grado versa sobre este tipo de redes, y su simulación mediante el simulador de redes Ns-3, que es la herramienta más utilizada en la comunidad científica para el desarrollo, experimentación y evaluación de estas tecnologías.

De esta forma, tras un estudio del campo de las redes de sensores subacuáticas y del funcionamiento del simulador Ns-3, se desarrollará un modelo para dicho simulador de los protocolos a implementar, derivando de ello un aprendizaje sobre dicho simulador. Conocer sus puntos fuertes y debilidades, crear escenarios de simulación y poder sacar sacar provecho de la aplicación son también objetivos deseados.

1.3 Estructura de la memoria

La memoria se organizará de la siguiente manera:

En el Capítulo 2 se explicará el estado del arte, con algunos de los algoritmos de encaminamiento para, posteriormente, comentar un resumen de las capas física, acceso al medio y de red de estos protocolos en las redes de sensores (con diferencias entre terrestres y subacuáticas).

Seguidamente, en el Capítulo 3 se introducirán las tecnologías empleadas, principalmente el simulador utilizado NS-3. Se comentará brevemente su arquitectura y las modificaciones que necesariamente se han debido llevar a cabo para poder realizar dicho trabajo. También se hará una breve descripción del proceso de simulación con generador de números aleatorios que incluye el simulador y por último, se describirán los motivos por los que se decidió pasar a desarrollar en el NS-3.

Después, en el Capítulo 4 se describirá los pasos para la implementación del protocolo DBR en el simulador, explicando las diferencias de implementación más importantes. También se detallarán las contribuciones al algoritmo derivadas de la implementación y prueba de la especificación original.

En el Capítulo 5 se muestran las evoluciones realizadas a partir de la implementación del protocolo DBR. En este caso se comenta el funcionamiento del protocolo clusterizado, así como sus principales métodos de funcionamiento y las clases que se compilarán en el simulador para su funcionamiento.

En el Capítulo 6 se exponen la experimentaciones realizadas, primero mediante la validación del DBR con las simulaciones citadas en [2] y [1]. Acto Seguido, se comparan entre ellas y con los resultados de la versión clusterizada.

En el Capítulo 7 se exponen las conclusiones generales del trabajo, así como la relación con estudios anteriores y con publicaciones.

CAPÍTULO 2
Estado del arte

2.1 Redes de sensores subacuáticos

Las redes subacuáticas tienen diferentes maneras de transmitir información, siendo algunas de ellas más eficientes que otras por el hecho del medio en el que transmiten, siendo diferente el efecto en agua que en el aire. Un ejemplo de esto pueden ser las ondas de radio que se utilizan en redes terrestres. Aquí es el agua lo que provoca que la atenuación de las ondas sea muy grande y por ello requieren mucha potencia de transmisión y en consecuencia antenas más potentes.

Este problema de la atenuación es de los más importantes que se encuentran en redes subacuáticas debido a varias causas. La primera de ellas es que la atenuación varía con la profundidad, a mayor profundidad mayor atenuación con lo que ya se tienen que tener en cuenta ciertos límites de profundidad. Lo mismo ocurre con la distancia y la frecuencia de transmisión, por tanto, se debe encontrar un equilibrio dentro de estas restricciones.

Estos problemas comentados antes, los solucionan las comunicaciones ópticas, que no sufren de atenuación y tienen un ancho de banda más amplio, pero por contra sí necesitan el agua limpia y cristalina, de lo contrario no existiría visión a través de ella y habrían problemas de dispersión de la señal.

A raíz de lo comentado anteriormente, el sistema de comunicación más usado en la actualidad en sensores subacuáticos es mediante ondas acústicas, que no obstante no es perfecto, dado que por ejemplo la atenuación también le afecta y hay que añadir, que al ser un sistema acústico, se ve afectado por el sonido ambiente marino (oleaje, cruceros...) que distorsiona la señal.

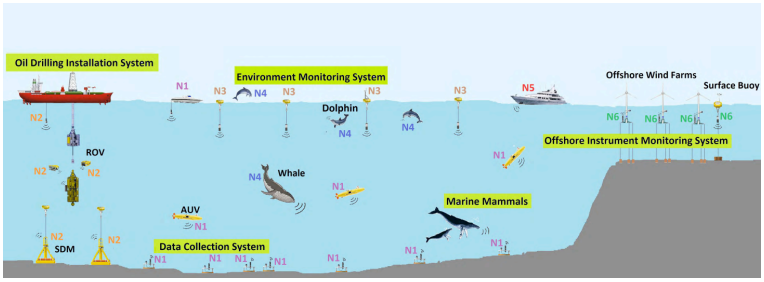

Figura 2.1: Algunos usos de las redes subacuáticas

Por último cabe destacar también dos características diferenciales entre comunicaciones terrestres y subacuáticas, porque con el agua, la velocidad de transmisión es mucho menor, teniendo en la superficie por ejemplo ondas de radio a una velocidad de propagación cercana a la luz mientras bajo el agua puede ser de 1000-2000 m/s. La segunda

característica es que, también debido al agua y todos los problemas que de ella derivan a la hora de transmitir sin todos problemas mencionados antes, que el coste de las redes subacuáticas se centra mas en la transmisión que en la recepción, siendo al contrario que en redes terrestres[10].

2.2 Capa física en redes subacuáticas

Como se explicará en el apartado siguiente, la tecnología que se utiliza en este trabajo como simulador es el NS-3 y la capa física que utiliza, en este caso, para redes subacuáticas, está implementada por el FuNLab de la Universidad de Washington.

Sin embargo esta configuración no es la única, hay investigaciones sobre otras modulaciones en redes subacuáticas y que también se utilizan actualmente. Un ejemplo de ello puede ser FSK, que es fácil de implementar y además muy eficiente en cuanto a energía. No obstante, no es perfecto y su ancho de banda es bajo. Quizás por ello, se han ido estudiando modulaciones más complejas como OFDM y combinaciones con MIMO, obteniendo resultados de hasta 12 Kbps a poca profundidad del agua.

2.3 Capa de acceso al medio en redes subacuáticas

La capa de control de acceso al medio (MAC) ha sido objeto de estudio a raíz de la existencia de redes subacuáticas[5]. En este punto también se aplican las mismas distinciones que en terrestres, pudiendo haber contención de tráfico por parte de los dispositivos o sensores que se comunican o no, dado que el problema radica en cómo repartir entre varios dispositivos, el uso de un único canal de comunicación o medio de transmisión, para que se puedan realizar diferentes comunicaciones al mismo tiempo.

En cuanto al control de la contención, uno de los protocolos más utilizados es el CSMA, que permite que múltiples estaciones compartan un mismo medio de transmisión haciendo que cada dispositivo indique su intención de transmitir antes de hacerlo para evitar colisiones entre los paquetes de datos, de forma el resto de dispositivos sabrán cuando hay colisiones y en lugar de transmitir la trama en cuanto el medio está libre, se espera un tiempo aleatorio adicional breve y solo si, tras ese intervalo el medio sigue libre, se procede a la transmisión (backoff). Pero CSMA también plantea problemas como el de los nodos ocultos (un dispositivo cree que el canal está libre, pero está ocupado por otro al que no escucha) o nodos expuestos (un dispositivo cree que el canal está ocupado, pero en realidad está libre pues el dispositivo al que escucha no le interferiría). Para solucionar estos problemas hay protocolos como el CSMA/CA o también MACA que usan el mecanismo RTS/CTS, donde previamente al envío se envían solicitudes y autorizaciones entre emisor y destino, donde todos los dispositivos al alcance del destino las escuchan.

Sin embargo, RTS/CTS para entornos subacuáticos no es perfecto, funciona mejor que un acceso aleatorio al medio por parte de los dispositivos, generalmente cuando las redes son grandes, con tamaño de paquetes grandes y con gran densidad de dispositivos, pero si la distancia de transmisión es demasiado grande , hay pocos nodos o el tamaño de paquete es pequeño, un acceso aleatorio al medio funcionaría igual o mejor.

Por otro lado, también hay protocolos libres de contención, como TDMA o FDMA En el primero consiste en ocupar un canal de transmisión a partir de distintas fuentes con un acceso múltiple por división de tiempo, donde el ancho de banda total del medio de transmisión es asignado a cada canal durante una fracción del tiempo total. Sin embargo, en una red con muchos nodos puede resultar impracticable por ser un protocolo centrali-

zado, mientras que FDMA no es aplicable a subacuático por el poco ancho de banda que tiene (apenas un pocos kHZ).

2.4 Resumen y clasificación de protocolos de encaminamiento

Hay diferentes variedades de protocolos de encaminamiento[6], tanto para redes terrestres como subacuáticas. Es importante conocer qué caracteriza a cada uno de ellos y en qué se diferencian para posteriormente entender en cual de ellos se engloba el cada uno de los protocolos desarrollados en esta memoria (ver ejemplos en 2.2).

La principal división entre protocolos se produce al ver como conservan o no la información de las rutas a los destinos. De esta forma existen en primer lugar, los protocolos reactivos, los cuales hacen que los nodos sólo guarden información de las rutas activas. Así, se obtiene la información de dichas rutas en el momento del envío del paquete y no antes, por lo que no hay una sobrecarga sobre si una ruta antes utilizada ahora es válida o no, simplemente se calcula el destino y se envía. El punto negativo es que conlleva una latencia extra, precisamente por este cálculo de la ruta antes del envío (con peticiones constantes hasta que se encuentra el destino). Los protocolos como el AODV son reactivos, el cual sólo guarda una entrada en la tabla de encaminamiento respecto al destino (por lo que se necesita recalcular el camino, en este caso mediante paquetes de descubrimiento Hello)[7]. También el que se va a implementar en esta memoria, el protocolo DBR pertenece a esta división[4]. Concretamente lo que hace DBR es mediante Broadcasting enviar paquetes desde la profundidad (es protocolo subacuático) y son los nodos intermedios los que reenvían a otros nodos (también mediante broadcasting) pero sólo si están a menor profundidad. En este sentido, no se guardan direcciones ni se realizan mensajes de descubrimiento. Más adelante se detallará su compartimiento con exactitud.

En segundo lugar encontramos los proactivos. Estos protocolos a diferencia de los anteriores mantienen en las tablas de encaminamiento las rutas que han sido usadas hacia los destinos por si se vuelven a repetir. Este tipo de protocolos tiene el inconveniente de que sus nodos han de encargarse de mantener actualizada estas tablas por si la red cambiase, esto implica envío de mensajes de actualización que provoca overhead a la red, teniendo a cambio, el cálculo de la ruta (si ha sido usada antes) hasta el destino de forma inmediata. Protocolos proactivos son como el DSDV que utilizan dos tipos de mensaje (de actualización de tablas completas y de actualización respecto al último de los anteriores).

Al margen de estas dos divisiones, se debe hacer referencia también a protocolos de redundancia o dirección común, como puede ser el protocolo CARP, que destaca porque permite a varios equipos de la misma red local compartir un conjunto de direcciones IP con el objetivo de ofrecer redundancia contra fallos en la puerta de enlace (aunque internamente si tienen una iP secundaria única). El ejemplo más utilizado de como se cumple esta función es con firewalls, cuando se tienen más de un equipo que realizan el filtrado de paquetes y uno cae, el otro puede seguir realizando la misma función y al tener la misma IP, el resto de equipos no se dan cuenta del error.

Protocolos que presentan importantes diferencias son también los geográficos. Los cuales como su nombre indica permiten que sus nodos conozcan su localización geográfica mediante una señal GPS (cosa que en redes terrestres es factible pero en subacuáticas su señal se distorsiona con el agua). Un ejemplo de ello es el protocolo GAF (Geographic Adaptive Fidelity). En este caso es un protocolo aplicado a redes para móviles adh-hoc pero también se puede utilizar con redes de nodos con sensores, dónde cada nodo usa su posición GPS. Aquí la red total se divide en trozos de red, donde el número de nodos es menor. Una vez dividida, en cada trozo los nodos van eligiendo cual de ellos está activo y

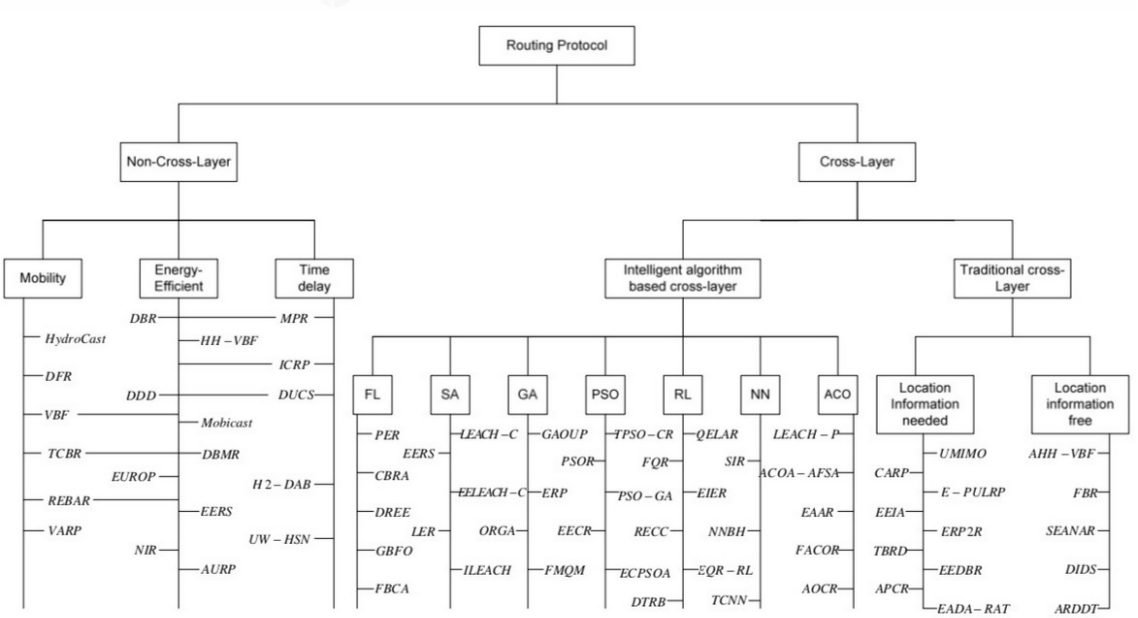

Figura 2.2: Esquema de protocolos

cual no o cual debe enviar y cual no mediante mensajes en los que se informan del tiempo que les queda de actividad, o también, de unión en dónde se agrupan entre ellos debido a su localización. De esta manera también se tienen una especie de pequeños clusters.

Por último y también en relación a los clusters comentados antes, se tienen protocolos jerárquicos, que son el objetivo a desarrollar en este documento, dado que evitan una sobrecarga de mensajes en la red (como pasaba en los proactivos) y a su vez abarcar la mayor área posible mediante clusters de nodos, cada cluster con un nodo líder o clusterhead que es al que todos los demás nodos de su área correspondiente transmitirán la información y éste posteriormente la procesará. Protocolos que entran en esta definición son LEACH (Low Energy Adaptive Clustering Hierarch) o EDETA (Energy-efficient Adaptative HiErarchical and RobusTA rchitecture)[9]. Los dos protocolos citados permiten a los nodos seleccionarse aleatoriamente como clusterhead, siendo estos los nodo que comprimen la información que les llega del resto de nodos (nodos hoja que recopilan la información y que pueden irse activando o desactivándose para ahorrar energía) y posteriormente entre estos clusterheads se creará un árbol hasta alcanzar al receptor o nodo sumidero, donde se vuelva toda la información. La información que envían los nodos a su clusterhead se realiza centralizadamente pero entre los clusterheads se envían los mensajes con técnicas como CDMA MAC para reducir las colisiones. Por ello realmente se ve que en estos protocolos no está toda la red interconectada entre sí, tan sólo diversos nodos líderes, evitándose problemas de anteriores protocolos con encaminamientos de red, descubrimientos, direcciones y congestión. Por esta razón en esta memoria se toma la idea de la implementación con clusters a partir del protocolo DBR (del cual se toma una base tanto para implementar como para comparar debido a su carencia de mensajes de configuración de la red).

CAPÍTULO 3
Tecnologías Empleadas

3.1 El Simulador NS-3

El simulador NS-3 es el simulador de redes de computadores con en que se va a proceder a implementar el nuevo protocolo y obtener los resultados que de él se derivan [3]. Es un simulador que permite realizar pruebas con dispositivos y aplicaciones reales. Para ello primero se va a describir sus características y posteriormente el proceso que sigue a la hora de realizar una simulación.

El simulador está implementado en el lenguaje C++, que fue elegido también porque permite compatibilidad con el código ya implementado en C, aunque también soporta simulaciones realizadas en scripts en Python. Esto se ha hecho pensando en la depuración de código (cuenta con una API basada en Python) y queriendo utilizar lenguajes modernos y usados en la actualidad.

NS-3 también ha evolucionado respecto a versiones antiguas como NS-2, donde no se tenía en cuenta el estándar de escritura de código, ahora se han intentado resolver esto, implementando un estándar más riguroso y un proceso de revisión de código. Otro problema en el cuál también ha mejorado respecto a versiones antiguas es en la dependencia entre modelos. Anteriormente si un modelo necesitaba de otros para funcionar, se editaba el código del modelo requerido, con lo cuál podría a modo de cadena, hacer que otros modelos dejaran de funcionar. En NS-3, se ha incorporado un modelo para agregar objetos, también en tiempo de ejecución y para saber si un objeto está agregado dentro de otro, pudiendo de este modo, hacer que un objeto nodo se pueda agregar otros objetos como un modelo de energía.

El sistema de trazas es otro punto que ha mejorado en el simulador. Ahora el sistema de trazas puede escribir su salida en diferentes ficheros de tipos ASCII o pcap. Por otro lado, el simulador no tiene un mecanismo de gestión de la memoria, lo cual es una desventaja, ya que al utilizar C++ no puede contar, por ejemplo, con el Garbage Collector de Java. Sin embargo, sí tiene una solución intermedia a este problema, y son los llamados punteros inteligente o Smart Pointers, que tienen un contador de referencias a un objeto y si estas referencias van desapareciendo, hasta llegar a 0, se libera la memoria.

Por último, cabe destacar el proceso de simulación que sigue el NS-3, dado que está compuesto por sucesos ordenados cronológicamente (y que también se pueden crear mientras avanza la simulación). La simulación acaba cuando no quedan más sucesos por simular o se llega a un estado límite, previamente configurado (un tiempo máximo).

3.2 Estructura interna del Simulador NS-3

El simulador está organizado en diferentes módulos según la figura 3.1. En el core o núcleo del simulador es donde están implementados algunas características comentadas en el apartado anterior como pueden ser los punteros, el sistema de trazas, los objetos o incluso el sistema de callbacks.

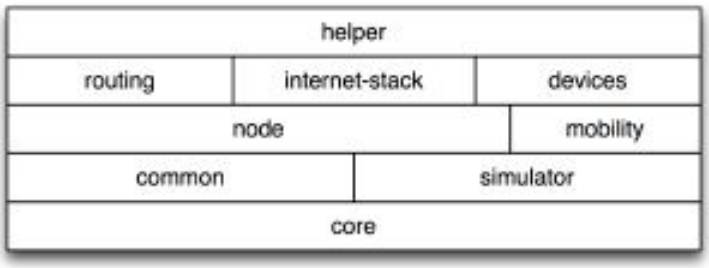

Figura 3.1: Esquema interno del simulador NS-3

La creación de paquetes, su estructura y su tratamiento se realiza en módulo Common y donde se planifican los sucesos de simulación (comentados en el apartado anterior) en el módulo Simulator. Por otro lado el módulo Node contiene la definición de la clase Node y más bloques como el NetDevice que son necesarios para la simulación de las redes y por últimmo en el módulo Mobility se encajan los modelos de movilidad, velocidad dirección o demás ajustes para conseguir que en una simulación los nodos tengan movimiento.

Los algoritmos de encaminamiento se encuentran en el módulo de de Routing, además del módulo Inernet-stack donde están los protocolos TCP, UDP, IPv4 e IPv6. Los dos últimos módulos que quedan son el Helper, que como su propio nombre indica, contiene ayuda (una API) para facilitar la programación de scripts a los usuarios y el módulo Devices, donde están los modelos de los dispositivos que permiten el acceso de los nodos a la red.

Una vez descrita la estructura se deben matizar diversos objetos que se han mencionado. Para empezar en el módulo Node, aquí se encuentra la clase Node, que no es más que un objeto que agrega más objetos. Esto es muy útil dentro del simulador ya que a estos objetos nodos se les pueden agregar movilidad para permitirles moverse por el entorno, sensores, antenas de transmisión, energía y demás características para construir sobre ellos toda un red.

El simulador también permite generar tráfico de datos, esto lo realiza mediante las llamadas .ªplicaciones". Éstas son objetos que simulan el tipo de tráfico con una aplicación real, si queremos un tráfico UDP debemos, por tanto utilizar la clase UdpEchoServer y configurar sus parámetros. El tráfico se complementa con el tipo de canal de transmisión, la clase çhannel", donde entre otros se puede configurar por ejemplo una Wi-fi mediante la clase WifiChannel.

Para terminar, se debe mencionar la clase NetDevice, que realiza la abstracción de todo el hardware de comunicación (como la tarjeta de red) o drivers y permite crear canales de comunicación.

3.3 Cambios en el Simulador NS-3

Para implementar el nuevo protocolo se han tenido modificar varios documentos o insertar nuevos ficheros, tal es el caso de la descripción de los nuevos nodos, como pueden ser el nodoDBR.cc, nodoDBR.h, headerDBR.cc y headerDBRh, que se comentarán y especificarán posteriormente.

3.4 Números aleatorios

El generador de números aleatorios utilizado por el simulador es MRG32k3a que tiene un periodo de 3.1 x 10e57 números aleatorios, una secuencia muy alta que se divide entre 1.5 x 10e19 streams de números independientes y cada uno de ellos se divide además entre 2.3 x 10e15 substreams con un período de 7.6 x 10e22

En este trabajo se van a utilizar conjunto a los objetos UniformRandomVariable, que admite la creación de objetos que devuelven números aleatorios desde una distribución uniforme fija. También es compatible con la generación de números aleatorios únicos a partir de varias distribuciones uniformes. Estos objetos se van a utilizar dentro del escenario de pruebas para situar los nodos dentro del escenario, así como para iniciar su velocidad o dirección:
"Direction", StringValue ("ns3::UniformRandomVariable[Min=0|Max=6.283185307]").

CAPÍTULO 4
Implementación del protocolo DBR

Para posteriormente implementar el protocolo basado en clustering, así como comparar resultados, se va a proceder a implementar un protocolo subacuático previo. Dicho protocolo está pensado también para ampliar el ecosistema de simulación y poder realizar estudios y evaluaciones de mecanismos de encaminamiento en redes subacuáticas. Este es el protocolo DBR, que, obtiene sus mejores prestaciones cuando hay una serie de nodos sumideros lo más cerca posible de la superficie y en el fondo una gran cantidad de otros nodos que retransmiten siempre de menos a mayor profundidad.

Este protocolo está catalogado como un protocolo reactivo (en referencia a las clasificaciones que hemos visto en apartados anteriores) dado que no se guardan informaciones de rutas previas, sino que cuando llega un paquete se decide «insitu» por donde direccionarlo.

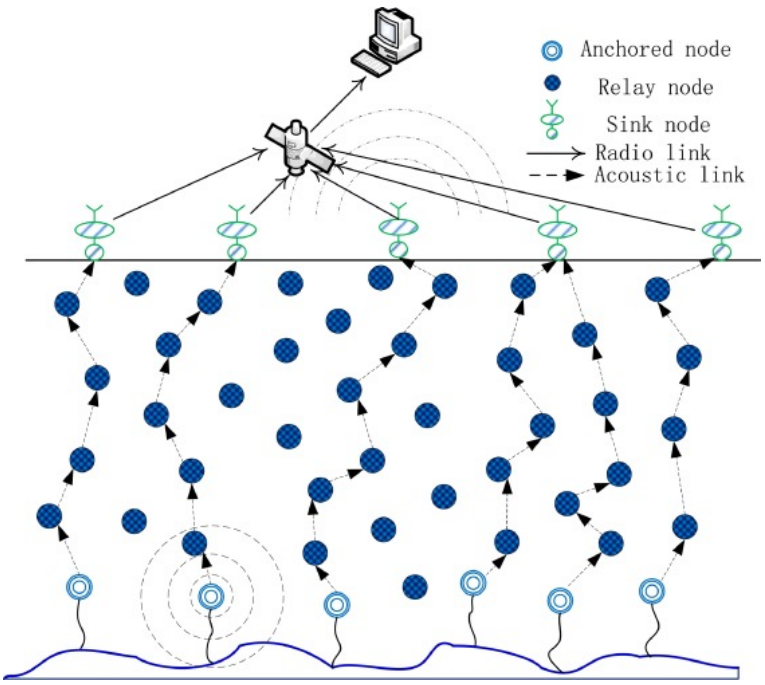

Figura 4.1: Proceso de envío del protocolo DBR

Como se puede ver en la figura 4.1. Su proceso es transmitir desde las profundidades hasta la superficie, por lo que cuando un nodo recibe un paquete, si este proviene de una

profundidad mayor podrá ser retransmitido. Si proviene de una profundidad menor se descartará al instante.

El protocolo funciona a base de Broadcasting (el destino son todos los que forman la red), con lo que todos los nodos que envían o retransmiten paquetes utilizan siempre la dirección de Broadcast, en ningún caso utilizan direcciones concretas a otros nodos. Esto por un lado alivia el tener que averiguar direcciones de los demás nodos y calcular destinos, lo cual provocaría cierta espera a la hora de transmitir, pero por otro lado permite que la red se congestione, porque si los nodos realizan siempre Broadcast, lo que se está haciendo es que todos o casi todos envíen a todos, que sin control de flujo provoca saturación en la red y pérdida de paquetes por posibles colisiones.

Con el fin de evitar estos problemas, algo que ayuda a ello es el hecho de que la profundidad afecte a la retransmisión de los paquetes, por ello aunque se realice broadcasting, si el nodo detecta, como se ha mencionado, que el paquete es de menor profundidad, no lo retransmite, ahorrando a la red el que ese paquete llegue a más nodos. Además como también se ha dicho, existe el ya de por si problema que ocurre en toda red: pérdida de paquetes por colisiones. Esto se intenta solucionar con dos modificaciones. La primera de ellas es creando una cola que contenga todos los paquetes reenviados y escuchados, de este modo aunque provenga el paquete de una mayor profundidad, si viene repetido no se retransmite. Por otro lado, la segunda modificación es la de no enviar los paquetes al instante que se reciben, es decir, cada nodo calcula en base a unos parámetros el momento en el cual va a enviar el paquete. Con esto se evita que dos o más nodos envíen a la misma vez y que los paquetes de éstos puedan colisionar. El esquema del funcionamiento se puede ver en la figura 4.2.

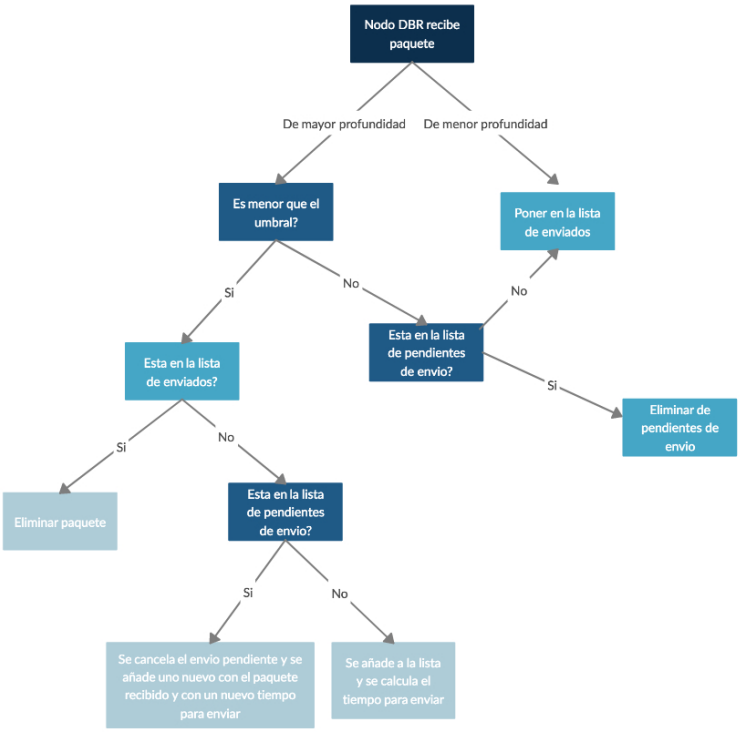

Figura 4.2: Funcionamiento DBR

Para poder implementar dicho protocolo se va a crear e instalar dentro del simulador, en la carpeta para simulaciones subacuáticas «Src/Uan/Model» dos nuevas clases: NodoDBR y HeaderDBR. La primera de ellas contendrá la definición y comportamiento de los nodos, tanto si son emisores, sumideros o intermedios. La segunda consistirá de la

definición de la cabecera de los paquetes que se inyectarán a la red y por tanto albergará la información necesaria para que los nodos retransmitan o no.

4.1 Implementación en el simulador del protocolo DBR

Una vez aclarado como funciona el protocolo, se ve ahora en detalle como está implementado en el simulador empezando por la clase cabecera (la cual se inserta en los paquetes de tipo ns3::Packet mediante el método «SetHeader» de la propia clase) headerDBR. Esta clase, igual que la siguiente, tienen tanto su «.h» como su «.cc» dónde se implementan sus métodos (ver diagrama de la figura 4.3).

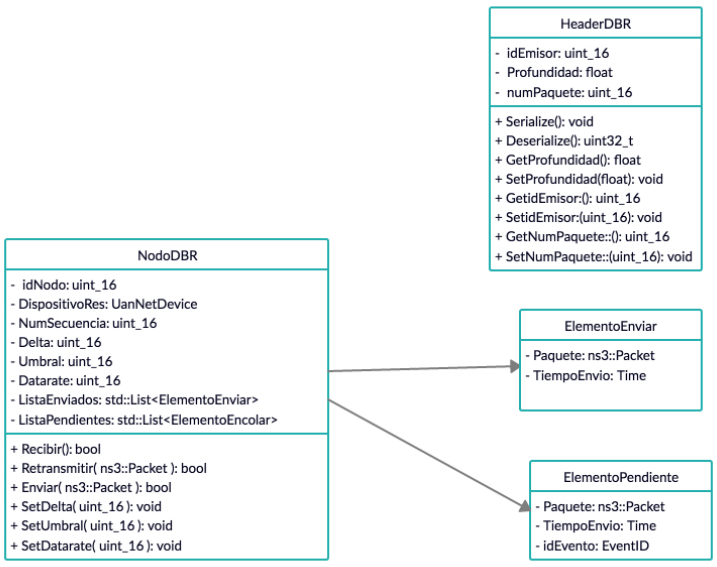

Figura 4.3: Diagrama clases DBR

La cabecera tiene tres atributos, necesarios para que los nodos reconozcan al paquete y de donde viene. Para empezar tenemos el id del nodo emisor del paquete, la profundidad del mismo y su número de paquete. Estas tres variables (el id y numero de paquete de tipo uint16, entero de 16 bits y la profundidad de tipo float) combinadas hacen al paquete identificable e imposible de que tenga un duplicado, cosa que en el nodo será determinante.

En el «.cc» tenemos los métodos para Serializar y Deserializar la cabecera del paquete (permiten leerla en los nodos) además de los genéricos «geters» y «seters» para cada atributo.

La clase nodoDBR es más compleja y con más métodos. Esta clase es la que definirá el comportamiento de los nodos, por lo que su importancia es capital. En primer lugar su «.h» contiene las definiciones de atributos y funciones que después se usan en el «.cc», entre ellas se destacan el dispositivo de red conectado al nodo, de tipo «UanNetDevice», que permite enviar y recibir paquetes del resto de la red. También incorpora el numero de secuencia (por si tiene que enviar, es el número que se pasa luego como número de paquete y que aumenta por cada envío), el id de nodo, así como las variables de su velocidad de transmisión (datarate) y las colas de tipo «std::list» para el almacenamiento de los paquetes recibidos y para los que están planificados para el envío.

Por último para poder calcular el instante en el que se reenvían los paquetes en cada nodo, hay dos atributos necesarios que se le pasarán al nodo cuando se cree, uno es el parámetro Delta y el otro el Umbral. El primero de ellos define la distancia máxima a la que se quiere enviar (el alcance) y el segundo indica la diferencia mínima de profundidad que se quiere entre nodo emisor y receptor (en el caso correcto de envío en el que el emisor esté a más profundidad que el receptor). La variable Delta se utiliza en la fórmula de la ecuación 4.1, que da como resultado el tiempo en el que se retransmitirá el paquete recibido para un alcance máximo (R) de envío de 100 metros (190db), una distancia al nodo origen del paquete (d) y un tiempo máximo de propagación τ.

Por otro lado, Umbral se utilizará como comprobante de que el paquete no viene de más profundidad de la que se especifica. Si esto se cumple entonces se accede al método y se completa la ecuación 4.1 calculándose el instante de envío.

$$f(d) = 2\tau/delta * (R - d), delta \in (0, R] \tag{4.1}$$

En el «.cc» de la clase nodoDBR, lo que se tiene son la implementación de los métodos para recibir, retransmitir y enviar paquetes. Para ello se tiene un método especial para recibir mensajes en el caso de ser sumidero dónde sólo se evalúa si los paquetes ya están en la cola de recibidos y dependiendo de ello, se acepta y se cuenta como recibido o un duplicado. El resto de nodos utilizan el método de recibir normal dónde cuando se recive un paquete, se evalúa la profundidad (que sea menor) del origen y que no estén ni pendiente de envió ni que ya haya sido recibidos.

Para finalizar queda el método para la retransmisión, que básicamente comprueba de nuevo que el paquete no esté pendiente de envío, calcula su tiempo para enviar (en caso positivo) y lo planifica invocando al método «Enviar()». Éste último, envía el paquete deseado mediante la dirección Broadcast invocando el método Send() del dispositivo de red «UanNetDevice» del nodo.

CAPÍTULO 5
Evolución al clustering

Como se ha explicado anteriormente y se ha confirmado con la implementación del protocolo DBR, en ningún momento se utilizan tablas de encaminamiento ni se hace uso de algoritmos ni mensajes para que los nodos tengan rutas de envío a otros nodos. Esto es así porque se utiliza siempre la dirección destino Broadcast, y se controla debido al descarte de los paquetes provenientes de menor profundidad que el Umbral, el tiempo de calculo para el reenvío y las colas para comprobar si hay repetidos.

Sin embargo, este tipo de implementación no parece la más óptima en cuanto al flujo de paquetes que circulan por la red ni respecto al consumo, ya que requiere que todos los nodos estén activos para reenviar a profundidades menores. Debido a esto se hace necesario encontrar una manera de solucionar estos problemas, evolucionando la implementación ya hecha y aprovechando sus mismas virtudes (carencia de configuración de tablas de encaminamiento en los nodos). Este es el motivo por el que se opta por los clusters (figura 5.1), conjuntos de nodos que se organizan en torno a un líder o maestro (llamado también ClusterHead) y que sólo tienen una dirección de envío única, tanto los nodos hojas que envían siempre a su nodo líder, como los propios nodos líderes que envían al nodo sumidero (el destinatario de los mensajes o la información a procesar) o a otro líder más próximo a ellos que el sumidero y que realiza la misma decisión de envío, creando un árbol.

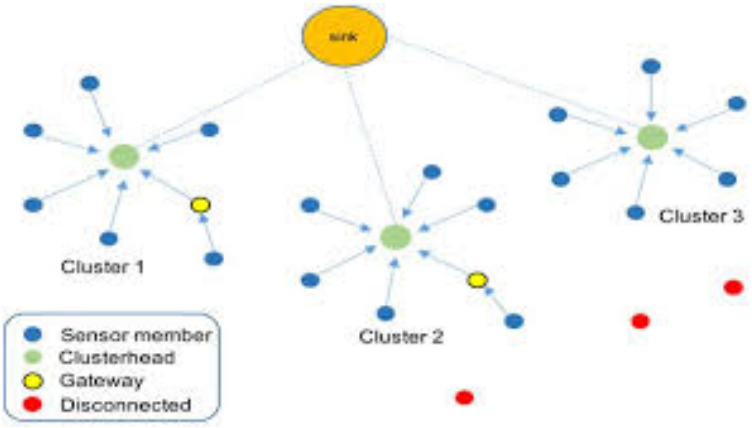

Figura 5.1: Ejemplo de Clustering

Para realizar la implementación cabe modificar y/o eliminar varios métodos y variables que se usaban con el protocolo DBR. Lo primero y primordial es eliminar tanto la variable Delta como Umbral, el cálculo del tiempo de reenvío, así como la sentencia que impedía retransmitir si venia de una profundidad menor. Ahora ya no son necesarias estas condiciones puesto que los nodos se pueden agrupar en clusters independientemente

de su profundidad y, al no emitir mensajes cuyo dirección destino es la de Broadcast, no hace falta el control sobre los reenvíos, porque tan sólo envían al nodo líder del cluster y éste a otro líder o al sumidero. Sin embargo para que se llegue a aplicar este procedimiento adecuadamente se necesita una primera fase de asociación y configuración del esquema de red, tanto para nodos líderes y sumideros, como para los clusters.

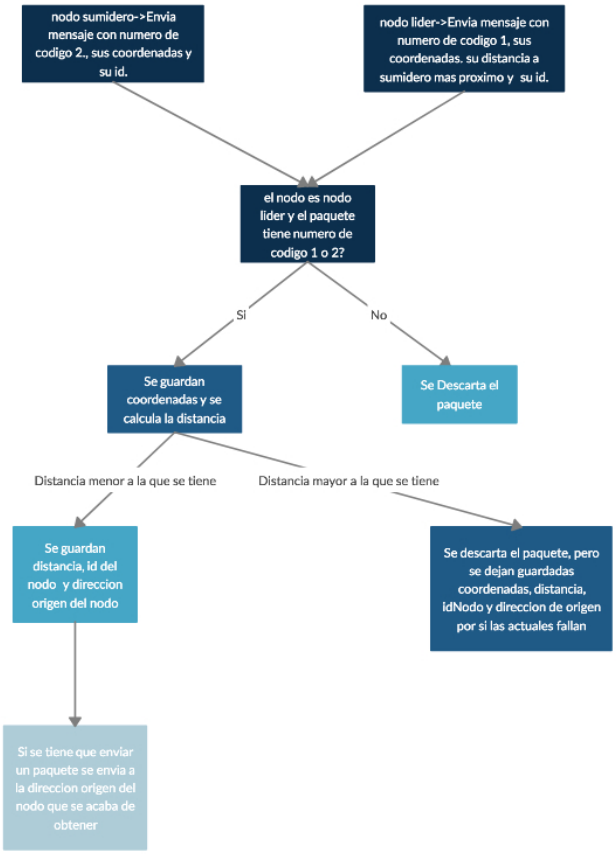

Figura 5.2: Proceso de configuración para nodos líderes

Esta primera fase de asociación tiene la misma estructura para nodos líderes como para nodos hoja, pero cambian los mensajes que reciben. Primeramente tenemos la asociación de los sumideros con los nodos líderes, éstos en el proceso de simulación de la red son los primeros en enviar sus mensajes de configuración mediante la dirección de Broadcast.

Envían sus ids de nodos, sus posiciones y el código de mensaje 1. Éstos mensajes tan solo pueden ser oídos y procesados por los nodos líderes y cuando los escuchan, calculan su distancia al sumidero, actualizan la distancia que tienen por defecto (1000 metros) y toman como dirección de envío a la dirección origen del paquete (la del sumidero). Este proceso en la simulación les lleva unos 5 segundos, hasta que reciben los mensajes y actualizan (una o más veces).

Después les toca el turno a los otros nodos líderes enviar el mismo mensaje con los mismos datos, salvo por el código, esta vez con código 2. El proceso es el mismo y si un nodo líder, al recibir un mensaje de otro líder, ve que la distancia de ese nodo líder a su sumidero más próximo es menor que la suya propia a su sumidero más próximo, la actualiza junto a la dirección de reenvío, que ahora será la del nodo líder que ha enviado el mensaje (véase figura 5.2). Sin embargo cabe destacar que tanto cuando actualizan o no,

los líderes se guardan todos los ids, direcciones y distancias que reciben en los mensajes, de esa manera si hay algún problema en la red tienen otras direcciones guardadas.

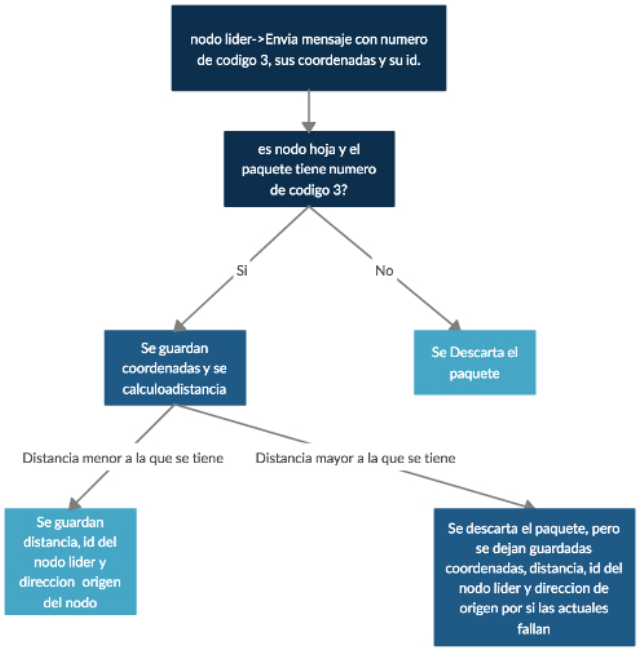

Figura 5.3: Proceso de configuración para nodos hojas

Después de que se tenga ya la estructura de envíos por parte de los nodos líderes (en la simulación tarda un lapso de otros 5 segundos), toca hacer la configuración con los nodos hojas. En este caso los líderes van a hacer exactamente lo mismo que anteriormente, pero con código de paquete 3. Sólo los nodos hojas reciben este paquete y calculan su nodo líder con el mismo proceso que los líderes con el sumidero. También cuando actualizan (porque un nodo líder está más cercano al receptor que otro) se guardan las direcciones, ids y distancias antiguas por si falla el nodo líder actual (figura 5.3).

5.1 Implementación en el simulador del clustering

Una vez se ha comentado el funcionamiento se debe ver como está implementado en el simulador. En este caso se van a tener el mismo número de clases que con el protocolo DBR. La clase «HeaderCluster» y la clase «NodeCluster» (figura 5.4).

La primera de ellas, al igual que con el protocolo DBR, es la cabecera que llevan todos los paquetes, aunque en este caso cobra aún más importancia porque dependiendo del tipo de código que lleve la cabecera servirá para configuración o como paquete normal de reenvío. Para ayudar en este aspecto la cabecera contiene el id del emisor (entero de 16 bits), un id del nodo Intermedio (entero de 16 bits) para saber si el nodo origen es el emisor, el numero de paquete y un atributo llamado «CodigoCluster» el cual toma valores de 0 (mensajes normales de envío), 1 (mensaje de configuración del sumidero), 2 (configuración del líder a otros líderes) y 3 (configuración del líder a nodos hojas). Tanto éste atributo como el id del nodo intermedio son enteros de 16 bits.

También dispone de un atributo double «distanciaSumidero» que se utiliza para cuando un nodo líder envíe su mensaje de configuración a otros nodos líder, pone este atributo el valor de la distancia de su sumidero más próximo, de esta manera cuando otro líder la

reciba, si ve que el emisor está más cerca a un sumidero que él, le reenviará los paquetes. Por último se tiene las coordenadas en tipo float. El total de tamaño que ahora tiene la cabecera es de 28 Bytes.

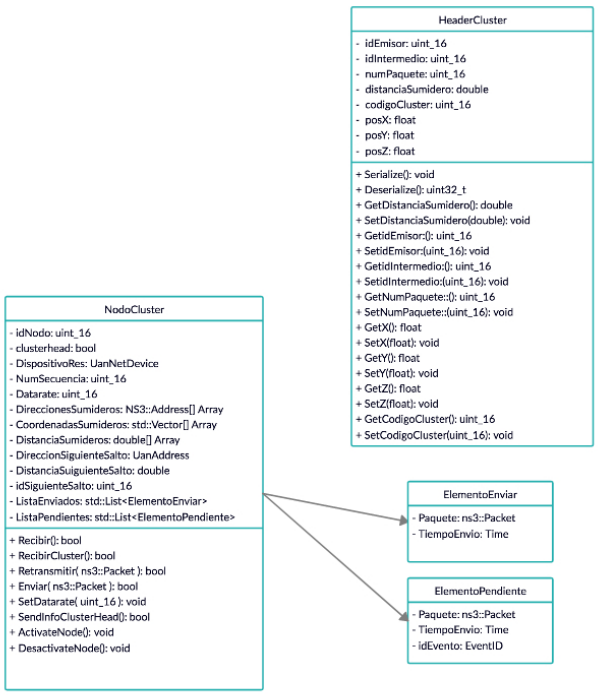

Figura 5.4: Diagrama de clases para la implementación de clusters

La clase NodoCluster realiza todas las funciones del protocolo. Ahora se le ha añadido un atributo booleano «clusterHead» el cual será verdadero si el nodo es nodo líder. Se calcula en cada nodo con la ecuación 5.1), donde c es el número de nodos líderes deseados, N el número de nodos y a es el valor temporal en el cual se calcula, si es al principio de la simulación es 1. Si T es mayor que un numero calculado al azar entre 0 y 1, el nodo es líder. Esta ecuación esta definida en [9].

Aparte, hay tres Arrays para guardar las direcciones de los sumideros, las coordenadas de los mismos y las distancias desde el nodo junto con otras dos variables llamadas «direcciónSiguienteSalto» de tipo «UanAddress» y «DistanciaSiguienteSalto» que son las que almacenan la dirección de reenvío (ya sea a otro líder o al sumidero) y la distancia al nodo al que se reenvía el mensaje y, que actúan, en el mecanismo de configuración (figura 5.2).

$$T(n) = (c/(|N| - 2c)) * a, n \in N \tag{5.1}$$

Los métodos que presenta esta clase son Recibir(), RecibirCluster(), Retransmitir() y Enviar(). El primero se asigna al dispositivo de red de los nodos hoja para que sólo esté activo en los mismos (mediante el dispositivo de red UanNetDevice y su función "SetReceiveCallback") de este modo sólo se activará cuando reciba un mensaje de un nodo líder de configuración para que el nodo hoja se una al cluster.

El segundo método se asigna sólo a los nodos líderes (del mismo modo que con el primero) y se activa al recibir un mensaje de configuración de los sumideros, de otros nodos líderes, al recibir un mensaje de un nodo hoja o de otro nodo líder que lo ha reenviado. En esto últimos casos se comprueba que no haya sido duplicado el paquete y que no esté ya planeado su envío con las listas de paquetes enviados y pendientes.

Ambos métodos para recibir se encargan de actualizar o calcular distancias según la ecuación de distancias punto a punto en un plano 3d 5.2. También llaman al tercer método, para reenviar el paquete (donde se pondrá en la lista de pendientes junto con un tiempo aleatorio de microsegundos y que se obtiene mediante una «ns3::UanRandomVariable») y éste llama al último método para que mediante la función Send() del dispositivo de red, el mensaje se envíe.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{5.2}$$

Para finalizar cabe destacar también los métodos para activar y desactivar el nodo (en caso de que sea un nodo suelto y no pertenezca a ningún cluster), para enviar mensajes de configuración (método SendInfoClusterhead()) o para establecer el datarate (velocidad de transmisionde datos).

CAPÍTULO 6
Experimentación

Se han efectuado varias campañas de simulación para la validación del protocolo DBR comparándolo en base a 1 solo nodo sumidero o 5 y posteriormente comparándolo con la implementación con clusters para ver si se obtienen beneficios o no.

La simulación con el protocolo DBR funciona con un nodo emisor situado en lo más profundo de la región inyectando paquetes cada segundo a la velocidad de transmisión (Datarate) que determine la simulación. Mientras, los nodos intermedios que reenvían dichos paquetes a la superficie se mueven a la velocidad que se disponga

Con la implementación con clusters la simulación es un poco diferente. No se tiene la necesidad de que el emisor esté en el fondo ni que sea sólo uno. Tampoco se transmiten los mensajes mediante Broadcast, sino que cada nodo hoja envía a su líder. Por tanto, cada segundo, se elige al azar un nodo hoja diferente de la región, y que además, tenga asignado un nodo líder dado que será el que envíe el paquete. Por ello, cada vez, se irán enviando mensajes desde diferentes clusters.

6.1 Validación del protocolo DBR con 1 nodo sumidero

Se ha efectuado teniendo en cuenta [2] con los parámetros de configuración del simulador que se disponen en la tabla 6.1.

Parámetro	**Valor**
Datarate	fijo a 10kbps
Frecuencia Central	12 kHz
Ancho de Banda	10 kHz
Modo	FSK
Modelo de error de paquetes	ns3 :: UanPhyPerNoCode
Modelo de la señal de ruido	ns3:: UanPhyCalcSinrDefault
Velocidad de Propagación acústica	1500 m/s
UAN Propagation Model	ns3 :: UanPropModelThorp
Modelo de dirección MAC	CWMAC
Modelo de Mobilidad	RandomWalk2Dmobilitymodel ->(speed: 4 m/s, directions,are choosen randomly)
Modelo de Energía	Acousticmodemenergymodel (TX: 50 W, RX/Idle:158 mW,,Sleep:5.8 mW)
Potencia de Transmisión	147 dB rem Pa
Potencia requeria para la adquisición de la señal SNR	10 dB rem Pa
Tamaño de Paquete	64 Bytes
Número de paquetes	60
Región de simulación	3D region of 1.5 x 1.5 x 1 (length x breadth x depth) km3
Número de Nodos	10–50, nodes are randomly deployed
1er Nodo sumidero en posición (1500, 1500, 0) m	
1er Nodo Origen en posición (0, 0, 1000) m	
Tiempo de Simulación	60 segundos
Delta y umbral (fijos)	100 y 0 respectivamente

Tabla 6.1: Configuración de la simulación de 1 sumidero

Este proceso de validación se ha hecho con un valor fijo de delta y umbral de 100 y 0 respectivamente, de ese modo nos aseguramos un rango máximo de retransmisión de paquetes (con Delta 100 ampliamos la distancia mínima de dos nodos candidatos a transmisión y con Umbral 0 el nodo retransmitirá todos los paquetes).

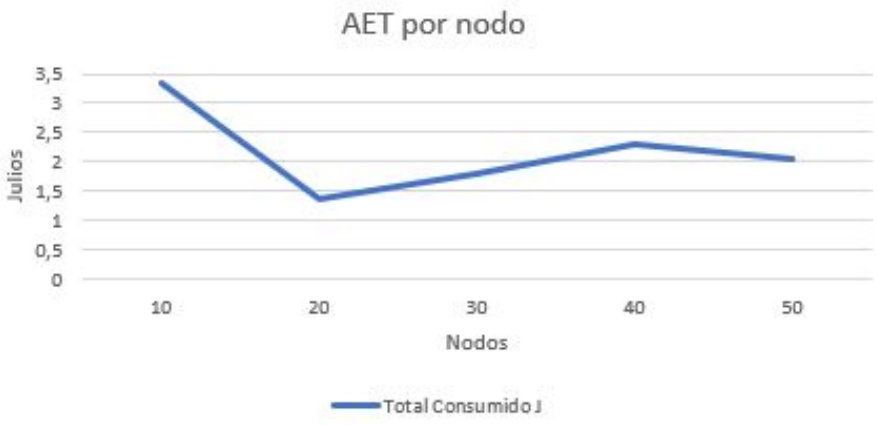

Figura 6.1: AET vs número de nodos

En la figura 6.1 se pueden ver los resultados de esta simulación. Se tiene Aet(Average energy tax) vs numero de nodos, esta fórmula contempla la energía media consumida por cada nodo y la divide entre los paquetes recibidos por el sumidero con éxito multiplicados por el numero total de nodos.

El resultado valida el modelo del documento, obteniendo primero una tendencia brusca a la baja con 10 nodos y posteriormente una subida a partir de 20. Esto es así porque los paquetes recibidos con éxito con 30, 40 o 50 nodos son mayores que con 20, lo cual hace que en el denominador de la fórmula de AET se obtenga un numero menor y si se divide por un numero menor, el resultado es mayor. Esta formula tiene en cuenta si el consumo se ha aprovechado o no, por tanto si hay más nodos en la red, debería haber más consumo, pero vemos que hay más consumo pero sin tanto éxito en la recepción de paquetes.

La relación de paquetes recibidos entre los paquetes enviados (PDR o Packet Delivery Ratio) lo vemos en la figura 6.2. Cuando se aumentan los nodos esta tasa se reduce, en parte por el también overhead que produce la red y que permite que haya pérdidas, sobretodo con gran cantidad de nodos, como puede ser 40 o 50 (donde parece que se estabiliza). Su pico, no obstante esta entre los 10 o 20 nodos donde más del 70 % de los paquetes se reciben.

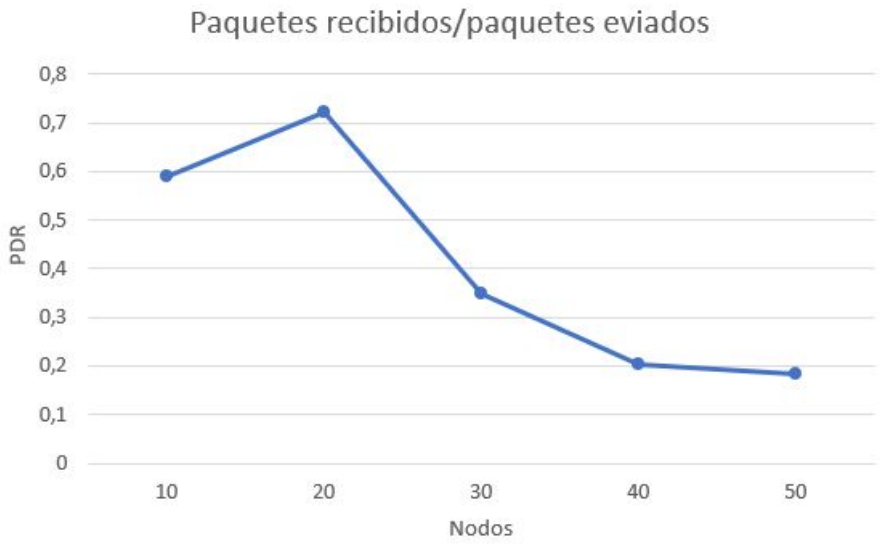

Figura 6.2: Packet Delivery Ratio (PDR) VS Número de nodos

6.2 Validación del protocolo DBR con 5 nodos sumidero

Esta parte de la experimentación se va a realizar teniendo en cuenta [1], dónde evalúa el correcto comportamiento del protocolo en base a 5 nodos sumidero. Esta vez la configuración del escenario de simulación va a ser diferente, no sólo por la ya citada diferencia entre nodos sumidero, también porque se va a cambiar el modelo de error, el modelo de movilidad, el modelo de ruido, la potencia de emisión de la señal, el tiempo de simulación y las simensiones del escenario entre otros (ver tabla 6.2).

Parámetro	**Valor**
Datarate	variable en las graficas (abajo especifico en unidades de bps)
Frecuencia Central	85 kHz
Ancho de Banda	1 kHz
Tipo de Modo	FSK
Modelo de error de paquetes	ns3 :: UanPhyPerGenDefault
Modelo de la señal de ruido	ns3:: UanPhyCalcSinrDefault
Velocidad de propagación acústica	1500 m/s
UAN Propagation Model	ns3 :: UanPropModelThorp
Modelo de dirección MAC	UANMACALOHA
Modelo de Mobilidad	GaussMarkovMobilityModel ->(speed: 4 m/s, directions,are choosen randomly)
Modelo de Energía	Acousticmodemenergymodel (TX: 2 W, RX: 0.1 W, Idle:0.01 W,,Sleep:0.008 W
Potencia de Transmisión	190 dB rem Pa
Tamaño del Paquete	variable en las graficas
Número de paquetes	variable en las graficas
Región de simulación	3D region of 0.5 x 0.5 x 0.5 (length x breadth x depth) km3
Número de nodos	200-300-400-600, nodes are randomly deployed
1er nodo sumidero en posición (250,250, 1) m	
2o nodo sumidero en posición (125,125, 1) m	
3ro nodo sumidero en posición (125,375, 1) m	
4o nodo sumidero en posición (375,125, 1) m	
5to nodo sumidero en posición (375,375, 1) m	
1er nodo sumidero en posición (250,250, 500) m	
Tiempo de Simulación	800 segundos
Delta y umbral (fijos)	variable en las grafica

Tabla 6.2: Configuración de la simulación de 5 sumideros

6.2.1. Resultados variando el parámetro Datarate

En esta sección se pone a prueba la tasa de paquetes recibidos (PDR) cuando se modifica el Datarate. Además de como afecta al mismo la variación del tamaño de paquete y

por qué. Se van a usar los siguientes parámetros de simulación fijos: paquetes de 64B (salvo en la figura 6.4), 800 paquetes durante 800 segundos (1 paquete/segundo), velocidad nodos 4 m/s, delta 100 y Umbral 20.

Para evaluar los resultados, se van a estudiar el PDR respecto a diferentes valores de Datarate para 200 nodos, PDR respecto a una variación de tamaño de paquete con 200 nodos y Datarate fijo de 1000 Bps y por último el consumo total respecto al aumento en los valores de Datarate.

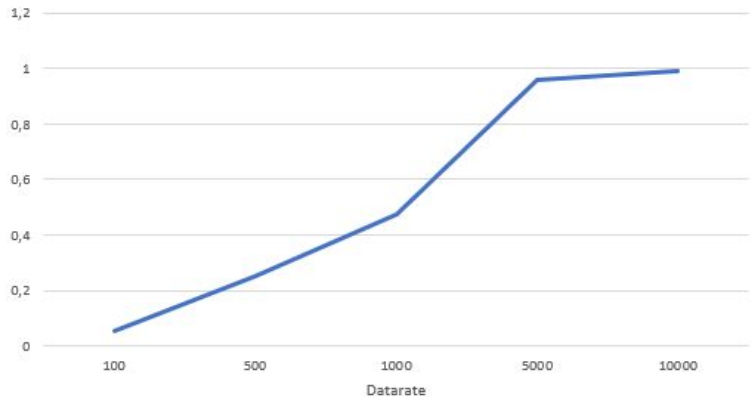

Figura 6.3: Packet Delivery Ratio (PDR) vs Datarate con 200 nodos

Como se puede observar en la figura 6.3, si se aumenta el datarate (que es la velocidad de transferencia de datos) aumenta a la vez el número de paquetes que se reciben. Esto es previsible dado que si se inyectan los paquetes a la red con más rapidez, se está reduciendo el tiempo en el que otro paquete puede colisionar con el que se está inyectando.

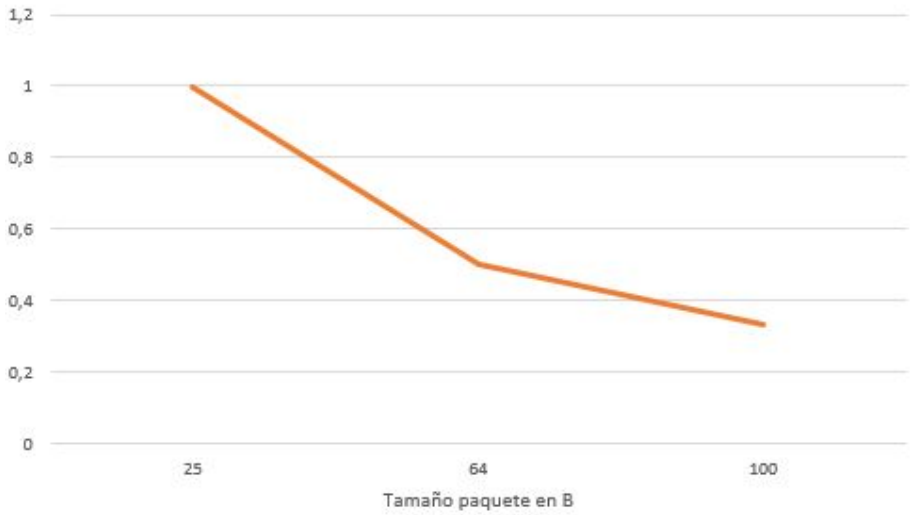

Figura 6.4: Packet Delivery Ratio (PDR) con Datarate 1000 vs tamaño de paquete con 200 nodos

También se puede observar como la tasa de recepción paquetes (PDR) se ve reducida si variamos su tamaño. En la figura 6.4 se muestra como se pasa de casi un 100 % a un 40 %. Al igual que se ha mencionado antes en cuánto a la importancia de que se transmitan rápido los paquetes para evitar colisiones, si se tiene un paquete mayor, por definición se tarda más en inyectarlo y en procesarlo, con lo que aumenta igualmente las probabilidades de colisión y debido a eso tendremos que volver a empezar la transmisión de nuevo y consumir más.

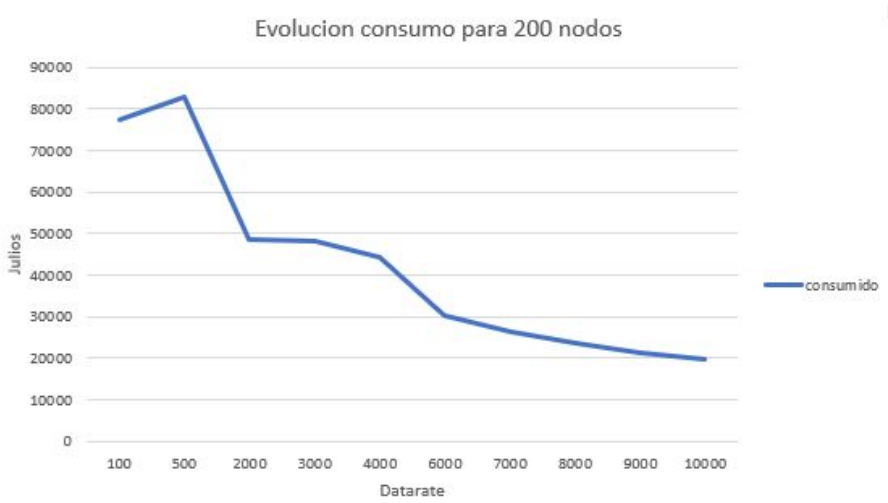

Figura 6.5: Datarate vs consumo total con 200 nodos

Es importante para esta sección, y también es necesario destacar, el consumo total de la red cuando se varía sólo el Datarate y con un número de nodos como es 200. Esto aparece en la figura 6.5 donde se obtiene una tendencia decreciente del consumo, ligada también otra vez al tiempo que se tarda en transmitir. En esta línea, si se tiene mayor potencia de transmisión y por tanto se está menos tiempo transmitiendo, el consumo que se invierte es menor que si se esta el doble o triple de tiempo. Además también ayuda a que con un Datarate mayor hay menos colisiones y el nodo no tiene que volver a empezar de nuevo el envío.

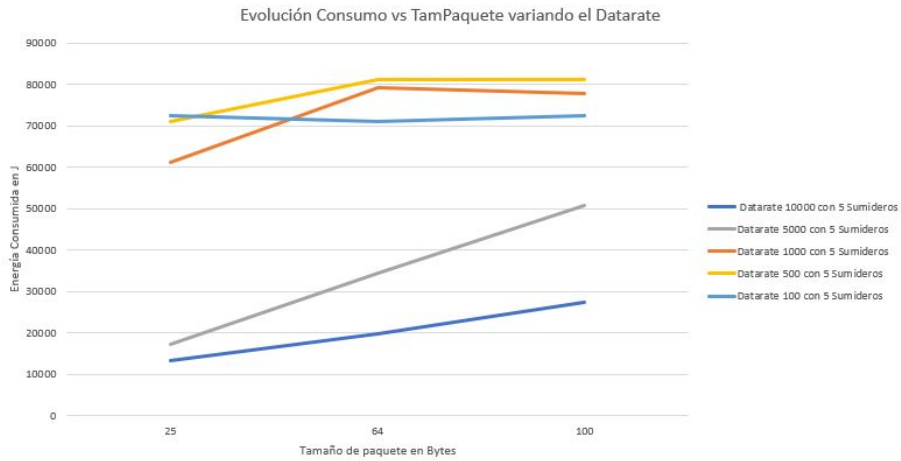

Figura 6.6: Consumo total con 200 nodos variando Datarate vs Tamaño de paquete

Por último a recalcar son los resultados de las figuras tanto 6.6 como 6.7. En ellas se ve primero como el consumo total de la red en general aumenta conforme aumenta el tamaño del paquete, pero que esto mismo es más pronunciado justo con los Datarates mayores, tanto 10000 como 5000 son los que menos consumen pero los que tienen la pendiente más significativa. No necesariamente debe ser deficiente, pero si que cuando tenemos velocidades altas de inyección de datos a la red se nota más cuando un paquete es más grande. Por otro lado también es lógico comentar que el consumo total de la red aumenta conforme se aumenta el número de nodos. Por ello a pesar de seguir la figura 6.7 la tendencia de menor consumo contra más datarate, el numero de nodos no cambia, estando siempre el orden de consumo 200 nodos <300 nodos <400 nodos <600 nodos.

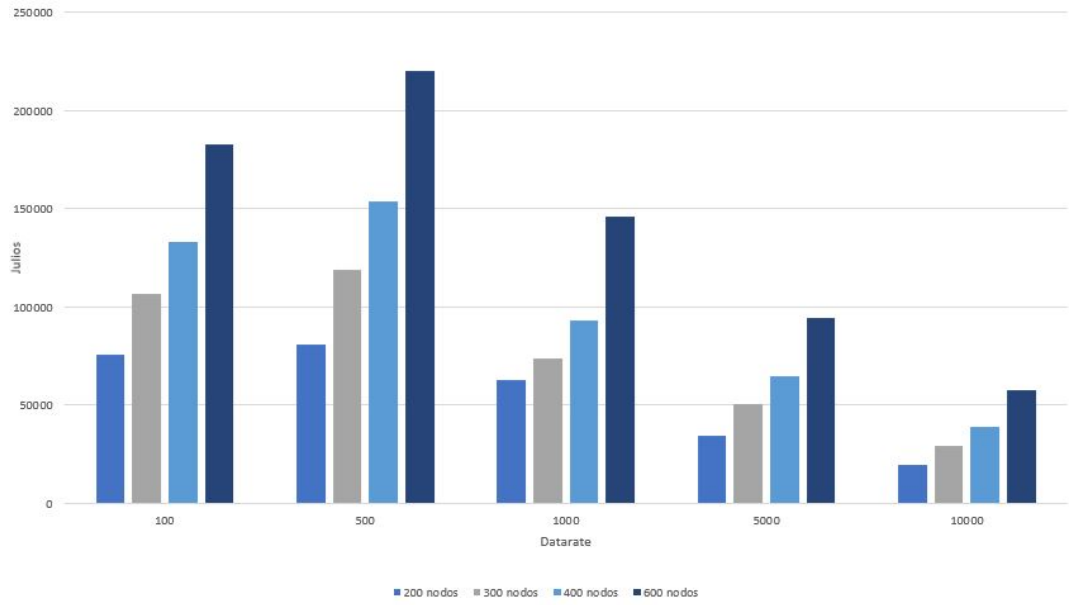

Figura 6.7: Consumo total variando numero nodos vs Datarate

6.2.2. Resultados variando el valor del parámetro Delta

No se debe olvidar que el valor Delta representa la distancia máxima a la que se es disponible enviar y/o recibir paquetes, esto obviamente afecta no sólo al tiempo de la ecuación del cálculo del tiempo de envío 4.1 sino también factores como el consumo, el retardo o la cantidad de paquetes que se reciben. También se van a usar los siguientes valores fijos : paquetes de 64B (salvo en la figura 6.4), 800 paquetes durante 800 sec (1 paquete segundo), velocidad nodos 4 m/s, Umbral 20 y Datarate 1000.

Para analizar este valor, se van a utilizar 3 valores de delta (100,50 y 25) aplicándolos a un número de nodos de 200, 300, 400 y 600. Con ello se pretende averiguar como reaccionará el consumo, el PDR, así como el retardo.

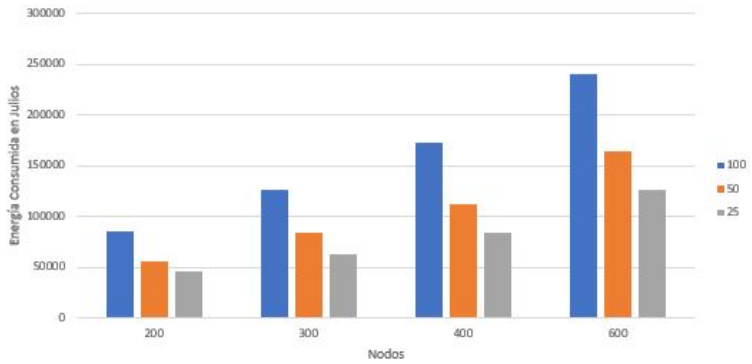

Figura 6.8: Consumo total variando numero nodos vs Delta

La primera figura que se va a comentar es la 6.8. En ella se ve claramente como con un valor de delta mayor, aumenta también más el consumo. La explicación de ello la tiene e la ecuación para calcular el tiempo de envío 4.1 donde Delta se encuentra en el denominador de la función, por ello a mayor delta siempre se tendrá un menor tiempo para enviar el paquete. Esto produce que los nodos reenvíen antes los paquetes y es más probable que se retransmita el mismo paquete (se recuerda que DBR funciona con Broadcasting) con lo que aumenta el consumo de energía.

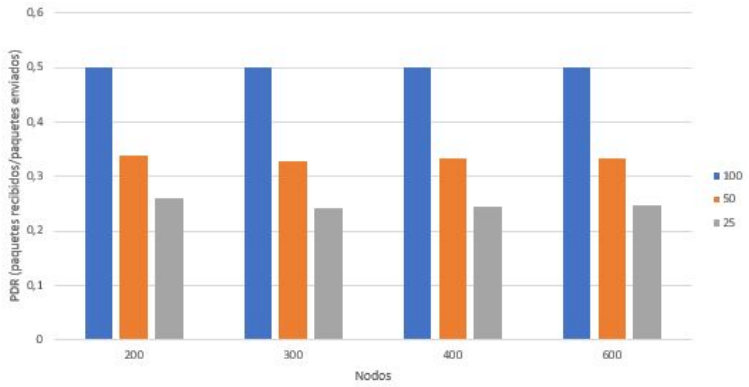

Figura 6.9: Packet Delivery Ratio (PDR) variando numero nodos vs Delta

En las figuras 6.9 y 6.10 se pueden también sacar conclusiones. La primera y evidente, es que el resultado del tanto PDR como del Retardo general de la red no varían significativamente con el numero de nodos (esto podría ser como anteriormente ha pasado por el valor bajo del datarate, ya que con valores de casi 5000 ya llega al 100 % (véase 6.3). Sin embargo, es suficiente para obtener una evaluación de que a mayor delta se obtiene una mejor tasa de recepción con Datarates bajos (dado que un mayor delta ayuda a retransmitir más), pero esta mayor retransmisión también puede generar más colisiones y aumentar el retraso.

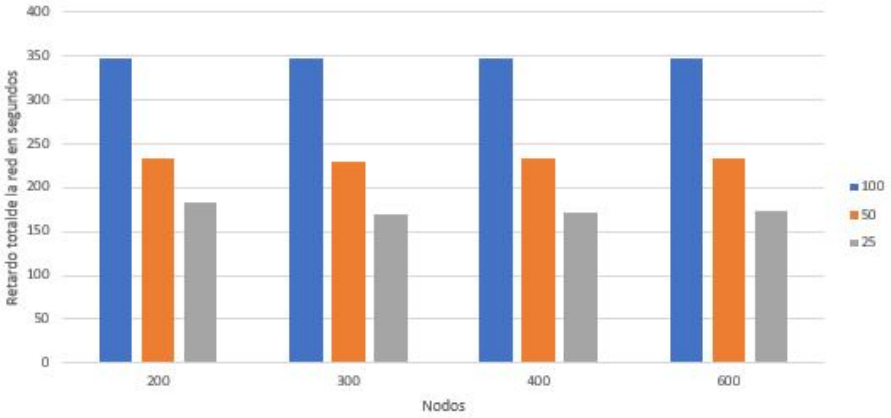

Figura 6.10: Retardo total en segundos variando numero nodos vs Delta

6.2.3. Resultados variando el valor del parámetro Umbral de profundidad

Este valor tiene en su finalidad poner el límite a la profundidad a la que se acepta un paquete, por lo que si el valor de umbral es mayor se aceptaran menos paquetes en los nodos dado que solamente serían aptos los paquetes que lleguen de los más profundo y la potencia de emisión de los nodos no sería lo suficientemente fuerte. Además también se van a usar los siguientes valores fijos : paquetes de 64B (salvo en la figura 6.4), 800 paquetes durante 800 segundos (1 paquete segundo), vel nodos 4 m/s, Delta 100 y Datarate 1000.

En este caso el parámetro se van a modificar utilizando 3 valores (20,40 y 80) aplicándolos como en la sección anterior a un número de nodos de 200, 300, 400 y 600. También se pretende saber como afectará al consumo, al PDR, así como al retardo.

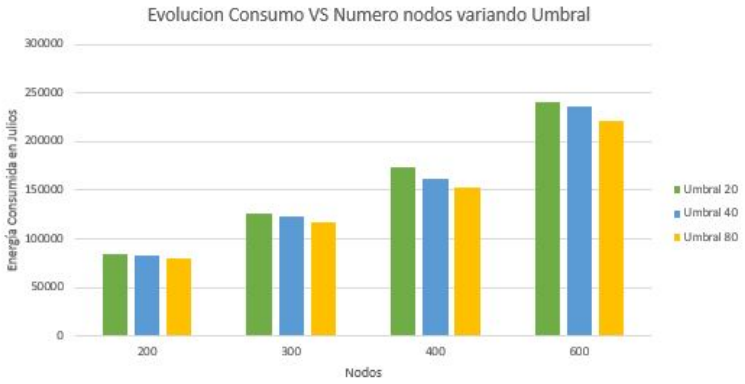

Figura 6.11: Consumo en segundos variando numero nodos vs Umbral

Como se ve en la figura 6.11 efectivamente se tiene que hay menor consumo con un umbral más alto. El resultado es lo que se ha explicado con anterioridad, si se establece un umbral más alto se está obligando al protocolo a sólo aceptar paquetes de más profundidad que ese umbral, arriesgándose a que los nodos más cercanos no puedan retransmitir y eliminando muchos caminos de envío. Por ello el consumo se reduce.

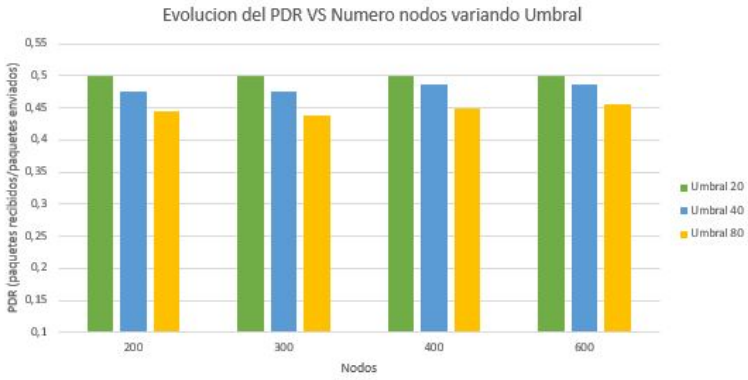

Figura 6.12: Packet Delivery Ratio (PDR) variando numero nodos vs Umbral

Las figuras 6.12 y 6.13 confirman lo analizado con el consumo. Dado que si se elimina la posibilidad de que los nodos más cercanos envíen sus paquetes, evidentemente tendrán que enviar los más lejanos, arriesgando que no tengan potencia suficiente de envío o que por el camino sufran colisiones o pérdidas. El PDR sigue esa tendencia, al igual que el Retardo. Con más nodos en la red, aun evitando esos caminos de envío, seguirá habiendo más posibilidad por la mayor cantidad de nodos restantes en una simulación con 600 que con 200.

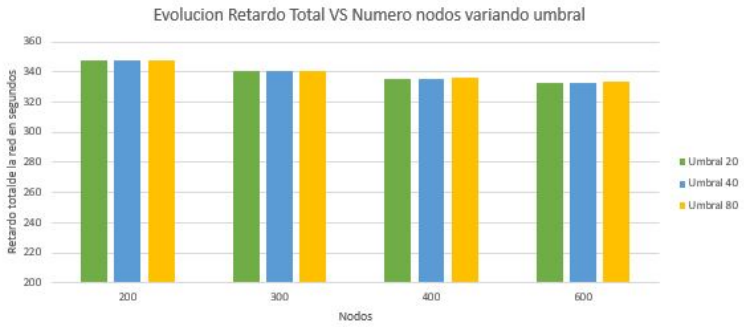

Figura 6.13: Retardo variando numero nodos vs Umbral

6.2.4. Resultados variando el valor del parámetro de Velocidad de movilidad de los nodos

La velocidad de movimiento de los nodos también es un parámetro a tener muy en cuenta, dado que si cambian de posición a mucha velocidad pueden provocar cambios en la estructura de la red haciendo que los paquetes que eran aceptados antes por provenir de un nodo a menor profundidad ahora ya no por el cambio de posición. Los valores fijos que se van a usar son: paquetes de 64B (salvo en la figura 6.4), 800 paquetes durante 800 segundos (1 paquete segundo), Delta 100, Umbral 20 y Datarate 1000.

Para este parámetro, al igual que con el Umbral y con Delta, aquí se van a simular 3 valores distintos de velocidad en m/s: 1, 4 y 15. Con ellos se verá como afectan al consumo, al retardo y al PDR de unos nodos que van de rango 200 a 600.

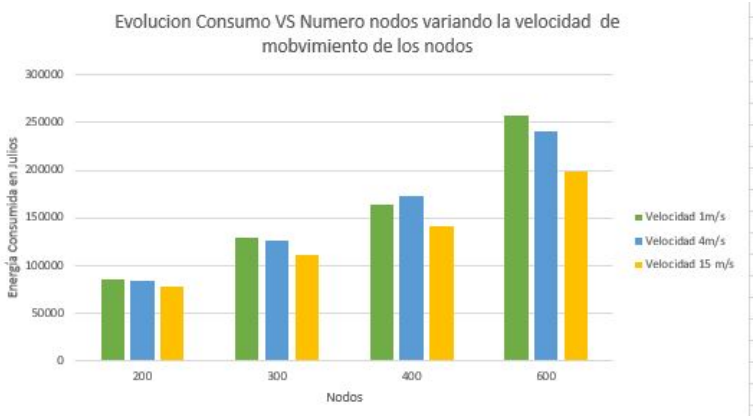

Figura 6.14: Consumo variando numero nodos vs Velocidad

La primera figura 6.14 muestra claramente como a medida que se aumentan los nodos, el consumo aumenta también, pero además a menor velocidad se produce un mayor consumo. Cuando hay una gran velocidad, la red se reestructura a menudo, con lo que muchos caminos de envío quedan rotos, al suceder esto muchos paquetes no se llegan a enviar ni a recibir, por lo que se pierde en actividad del nodo. Se corrobora cuando se ve la figura 6.15, y se observa que la tasa de paquetes recibidos es siempre mayor con una velocidad menor, es decir con una red más estable. Además la diferencia entre esta tasa se va reduciendo a medida que va aumentando el número de nodos, dado que es más posible que un camino exista.

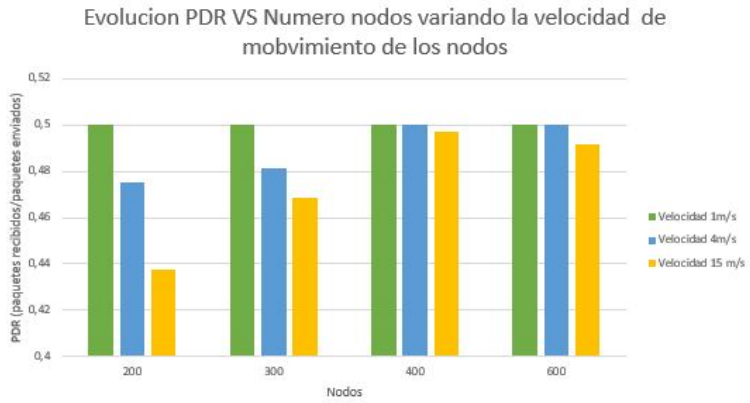

Figura 6.15: Packet Delivery Ratio (PDR) variando numero nodos vs Velocidad

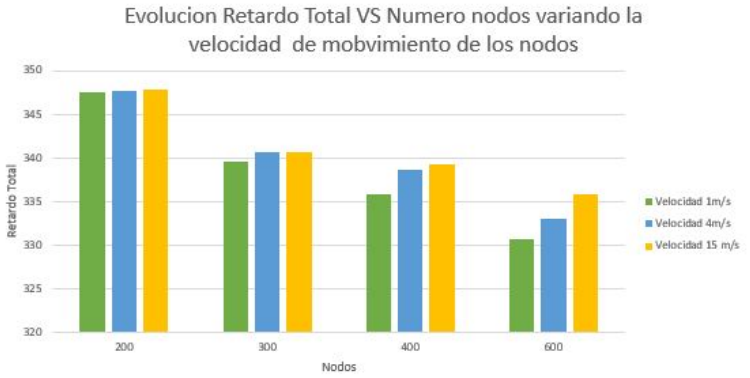

Figura 6.16: Retardo en segundos variando numero nodos vs Velocidad

El retardo de propagación total de la red también se ve afectado por la velocidad (figura 6.16) y se le puede aplicar la misma explicación que en las anteriores figuras, donde con un mayor numero de nodos, es más fácil encontrar caminos hacia el destino y por tanto no hay tantas pérdidas o colisiones.

6.3 Comparación del protocolo DBR con 1 VS 5 nodos sumidero

En este apartado de la experimentación ya se han comprobado, razonado y explicado los resultados obtenidos tanto con un nodo sumidero como con 5. Ahora pues la idea es compararlos y ver en qué se diferencian y debido a qué lo hacen exclusivamente. Para ello hay que poner en igualdad de condiciones la simulación dado que en los experimentos que se han hecho hasta ahora tenían características diferentes como el modelo de movilidad o el de error. En este aspecto dado que ha sido con la version de 5 nodos sumidero con la que se ha hecho un mejor y más profundo análisis del protocolo, se utilizaran sus configuraciones (tabla 6.2). Aunque hay un conjunto de parámetros, que salvo que se especifique en la figura correspondiente (porqué es el parámetro a medir) son fijos: paquetes de 64B, 800 paquestes durante 800 segundos (1 paquete segundo), datarate 1000, vel nodos 4 m/s, delta 100 y Umbral 20

Lo que resulta de mayor interés a comparar siempre son las prestaciones de un protocolo respecto a la tasa de paquetes recibidos (PDR) y al consumo. En esa línea se obtienen las siguientes gráficas también valorando el factor Delta.

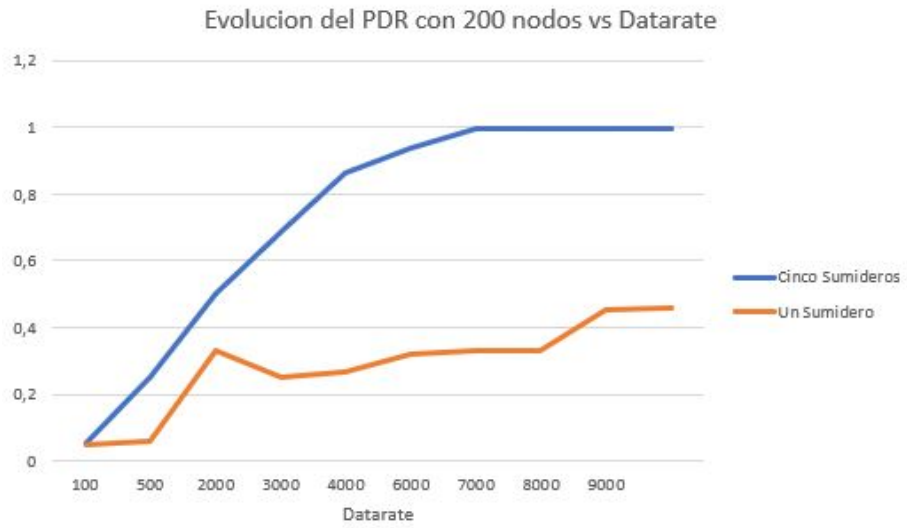

Figura 6.17: Packet Delivery Ratio (PDR) vs Datarate con 200 nodos

Los primeros resultados se obtienen mediante la figura 6.17 donde se aprecia que claramente sin importar el Datarate utilizado, 5 nodos sumidero siempre van a tener una tasa de recepción de paquetes superior por pura matemática: si son más de uno los que reciben, la red está más equilibrada y el tráfico repartido, evitándose fenómenos como el Overhead o el desborde de la cola de paquetes recibidos. Además, el hecho de que sólo haya un nodo sumidero en los niveles superiores (donde se está más cerca de la superficie) lo deja como única opción de envío y si no es un nodo, sino 20 o 30 nodos los que realizan este envío al sumidero, puede haber una perdida enorme de paquetes (en Datarate 900 hay casi un 50 % de pérdida). par

En la figura 6.18 también se aprecia esta diferencia, sin importar el número de Bytes que ocupe el paquete (ver figura 6.5 y su explicación) y guarda relación con el grave problema de la congestión que tiene la red con sólo un sumidero.

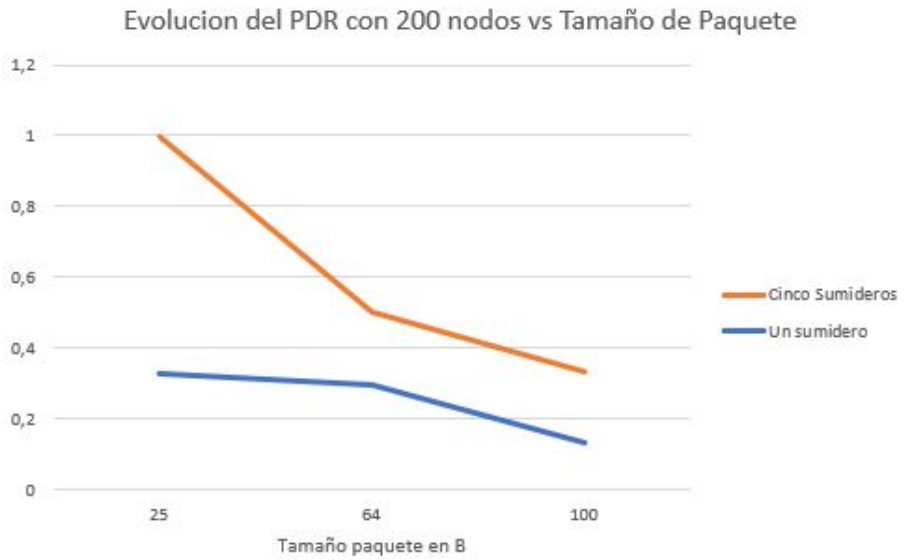

Figura 6.18: Tamaño de paquete vs Datarate con 200 nodos

Junto tras la tasa de paquetes recibidos, se acompaña siempre el consumo total. En el caso a analizar, variando el rango de Datarate y para hacer la figura 6.19 más completa, también que cada barra sea un cantidad de nodos diferente como se indica en su leyenda.

De ese modo se tiene la energía total consumida para 200, 300, 400 y 600 nodos (con 1 o con 5 sumideros) por el rango de Datarate de 100 a 10000. El resultado, como cabe esperar esta liderado por el consumo de los nodos con 1 sumidero. Tan sólo basta comparar el consumo de 600 nodos con 5 sumideros (el que más consume) para saber que con Datarate 1000, 5000 y 10000 está por debajo en consumo de todos los nodos que tienen 1 sumidero excepto menos el de 100 nodos. Esto se debe porque, aunque a medida que el Datarate aumenta, el consumo disminuye, pero la diferencia entre el consumo de los nodos con 5 sumideros y los de 1 se hace más grande, por lo que van a haber más nodos retransmitiendo paquetes, y si también se sabe que al tener 1 sumidero hay un efecto çuello de botella", pues van a haber más colisiones e intentos de retransmisión.

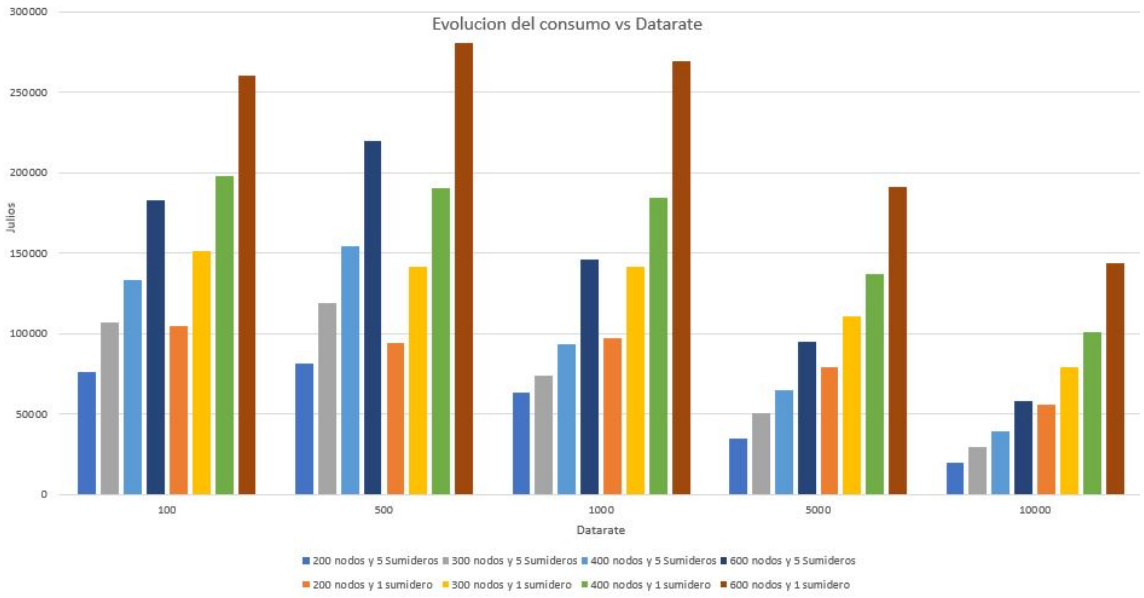

Figura 6.19: Consumo total por nodo VS Datarate

6.3.1. Comparación teniendo en cuenta la variación del parámetro Delta

También resulta interesante comparar como afecta el valor Delta, después a haber visto como es el valor que más afecta al comportamiento de la red en la sección 6.2.2. Por ello se va a ver el retardo (esencial para entender si afecta el çuello de botella"que hay con un sumidero), el consumo y también el PDR.

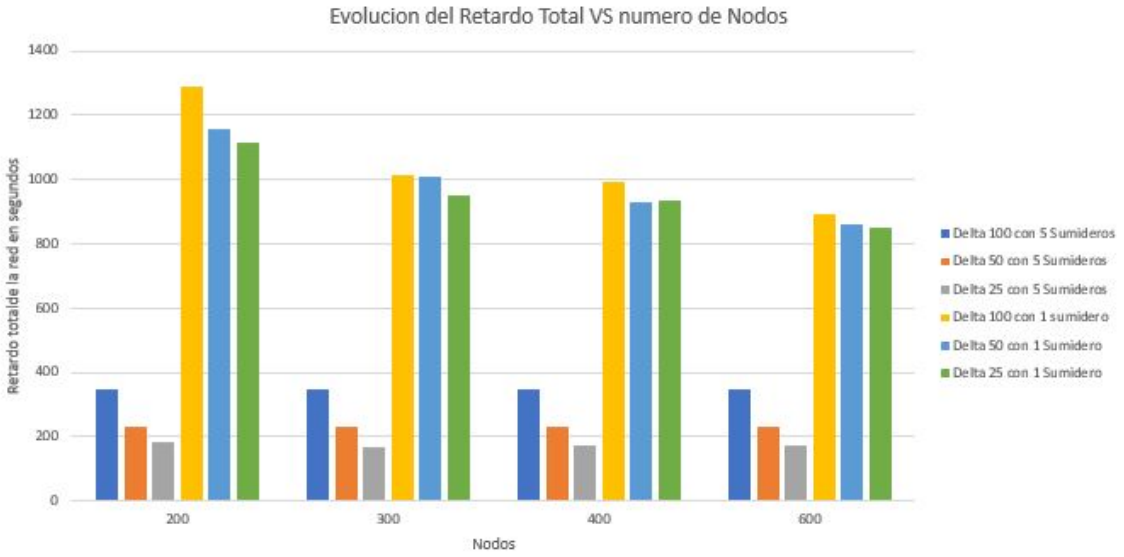

Figura 6.20: Retardo según Delta VS Número de nodos

Como también era lógico suponer, el retardo se ve gravemente afectado cuando sólo hay un nodo sumidero. No solo disminuye con un Delta menor como se ha explicado, sino que disminuye drásticamente con 5 sumideros, no importa el número de nodos a los que se esté evaluando (aunque siguiendo la tendencia descendente de Delta con la cantidad de nodos estudiado en 6.2.2). La figura 6.20 muestra como la red se congestiona cuando los nodos tienen en la capa menos profunda, sólo un destino.

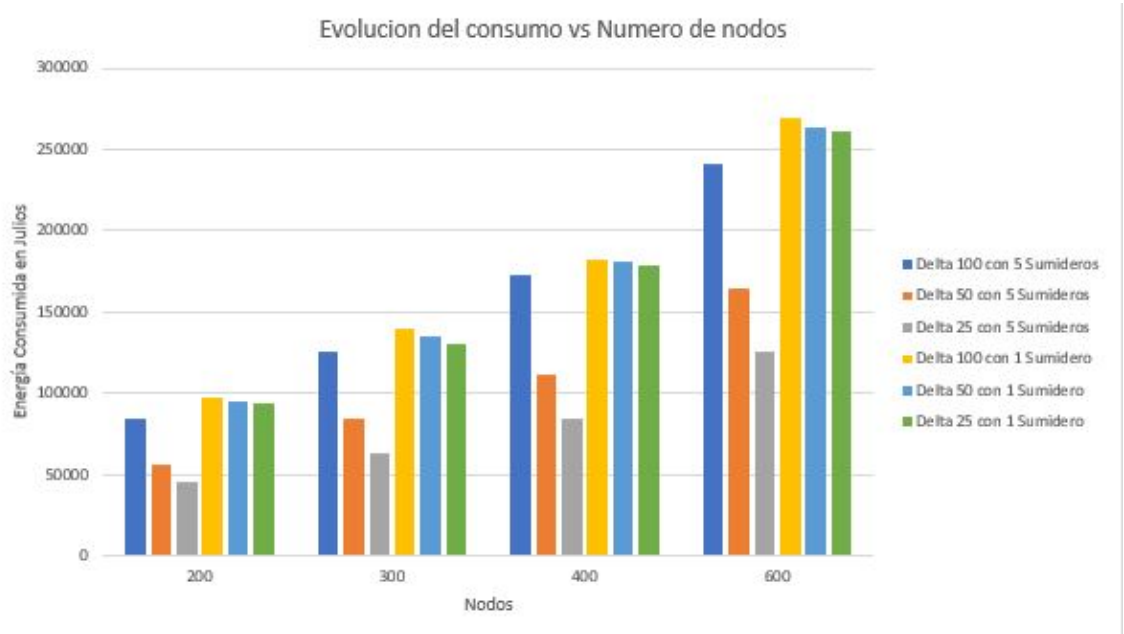

Figura 6.21: Consumo según Delta VS Número de nodos

También en el consumo según delta sale una comparativa negativa respecto a un solo sumidero, al igual que la tasa de paquetes recibidos. Ambas figuras 6.21 como 6.22 siguen las tendencias de Delta vistas, las simulaciones con un sumidero tienden igual, pero de una manera mucho más superflua. En cualquier caso el resultado es similar a las comentadas figuras 6.17 y 6.19.

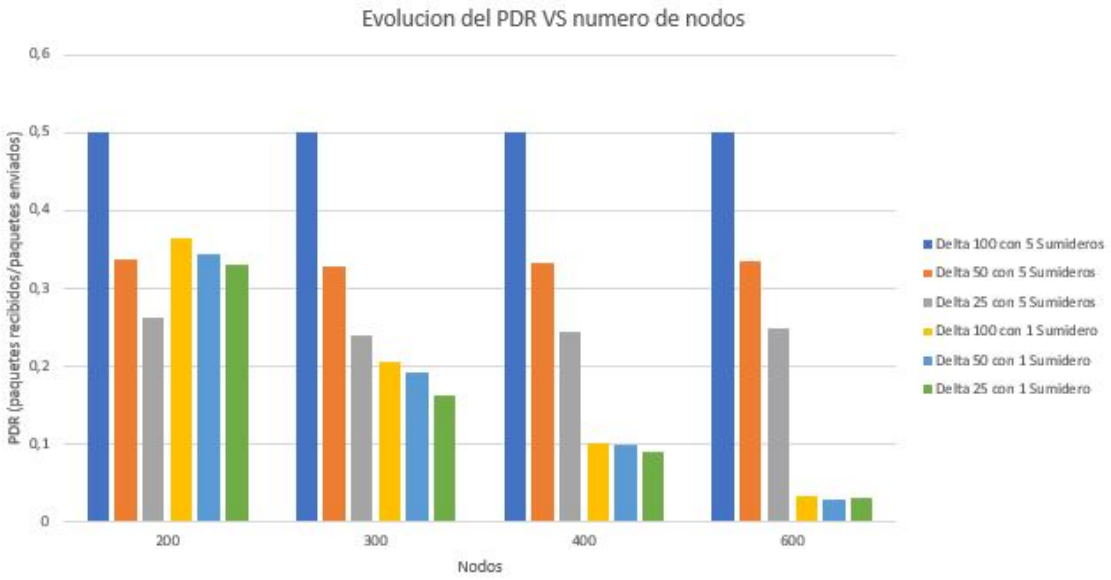

Figura 6.22: Packet Delivery Ratio (PDR) según Delta VS Número de nodos

6.4 Resultados del protocolo con implementación clusterizada

En este caso se usarán los mismos parámetros del simulador utilizados en la tabla 6.2. Sin embargo se obtiene una diferencia importante: la movilidad de los nodos. Dado que es imperativo que se organicen en clusters, necesitan configurarse y agruparse en torno a aquellos nodos más cercanos, así como establecer un árbol de nodos líderes que desembocará en el nodo sumidero. Esto no sería muy factible si, con la movilidad, los nodos se agruparan y desagruparan constantemente. Por ello se ha optado por usar un modelo de movilidad constante, donde los nodos se sitúan aleatoriamente por toda la región de simulación pero posteriormente, se quedan fijos. El modelo es: «ns3::ConstantPositionMobilityModel».

Aprovechando que la idea es la creación de clusters, el objetivo pues es ver como la cantidad de clusters que se crean afectan al consumo y a la tasa de paquetes recibidos (PDR).

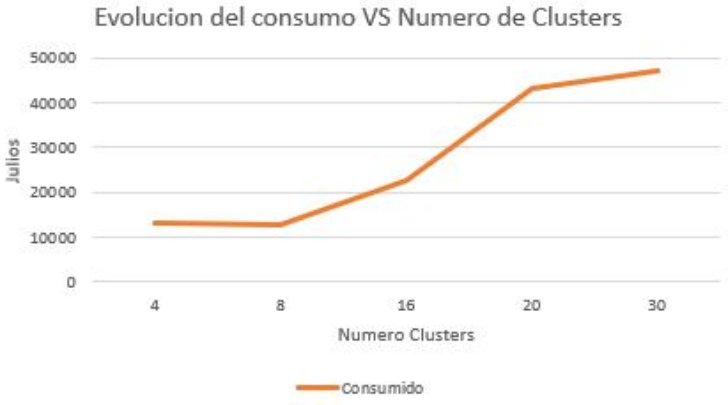

Figura 6.23: Consumo VS Número de Clusters

Como se muestra en la figura 6.23 a medida que se van creado más clusters, se va aumentando el consumo. Esto tiene lógica ya que contra más clusters se creen más nodos

líderes habrán que tendrán que reenviar paquetes que reciban. Por ello los caminos hacia el sumidero tendrán más saltos (hay más nodos líderes y éstos retransmiten su paquete a otro líder hasta llegar al sumidero) y por lo que más nodos participan en el trayecto de un mensaje a su destino y se consume más.

Por otro lado la figura 6.24 muestra que la tasa de recepción de paquetes no aumenta con el número de clusters. No hay una disminución grande pero sí lo suficiente para ratificar que si con más clusters, en el camino de un paquete a su destino participan más nodos líderes, es bastante probable que estos líderes también quieran transmitir, provocando tráfico y colisiones. Los resultados de ambas figuras están simulados con un datarate de 2000bps, tamaño de paquete 64B y un envío de 800 paquetes.

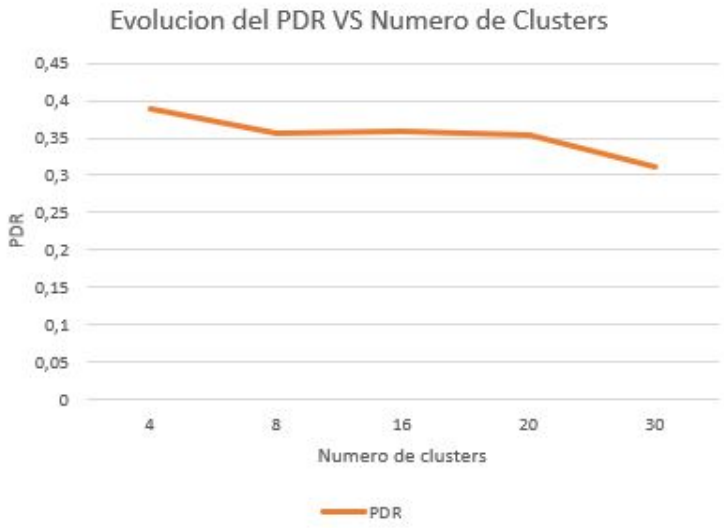

Figura 6.24: Packet Delivery Ratio (PDR) VS Número de Clusters

6.5 Comparación de resultados general

En esta sección se van a mostrar simplemente la comparativa de las tres tipos de implementaciones que se han simulado en este trabajo: DBR con 1 nodo sumidero, DBR con 5 nodos sumidero y clustering (en este caso con 4 clústers). Lo primordial es saber quién es más eficiente obteniendo resultados en consumo y en tasa de paquetes recibidos, así como también consumo dependiendo de la velocidad de tranmisión de datos (Datarate).

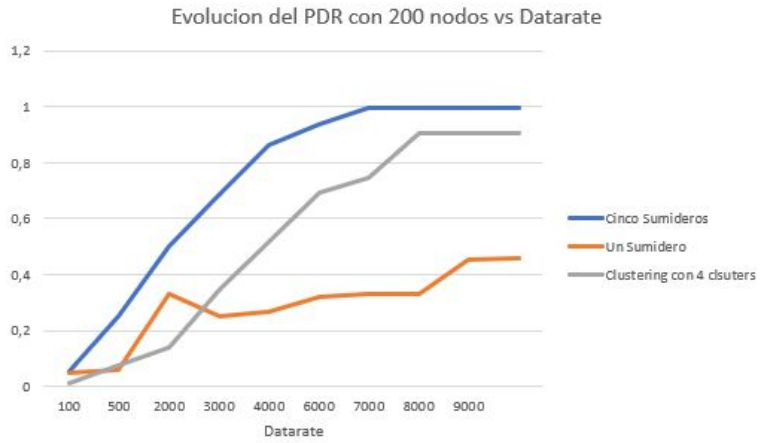

Figura 6.25: Packet Delivery Ratio (PDR) VS Datarate

La figura6.25 muestra un resultado algo sorprendente. La evolución del PDR de la implementación con clusters es muy parecida a la del DBR con 5 sumideros. Empieza con poca con eficacia a Datarates muy bajos pero una vez pasa del valor de 2000, la diferencia con el DBR de un sumidero es importante, de tal forma que llega a un total de 80 % de paquetes. Eso sí, sigue estando un poco por debajo de los resultados de DBR con 5 sumideros, al fin y al cabo no se debe olvidar que en clustering también se crea un cierto cuello de botella cuando dos o mas nodos líderes reenvían a un nodo líder concreto o a un sumidero. No obstante no es lo mismo el envío a un nodo líder por parte otros dos que con la implementación de DBR con 1 sumidero donde puede haber 20 o más nodos cercanos a la superficie enviando a 1 sumidero.

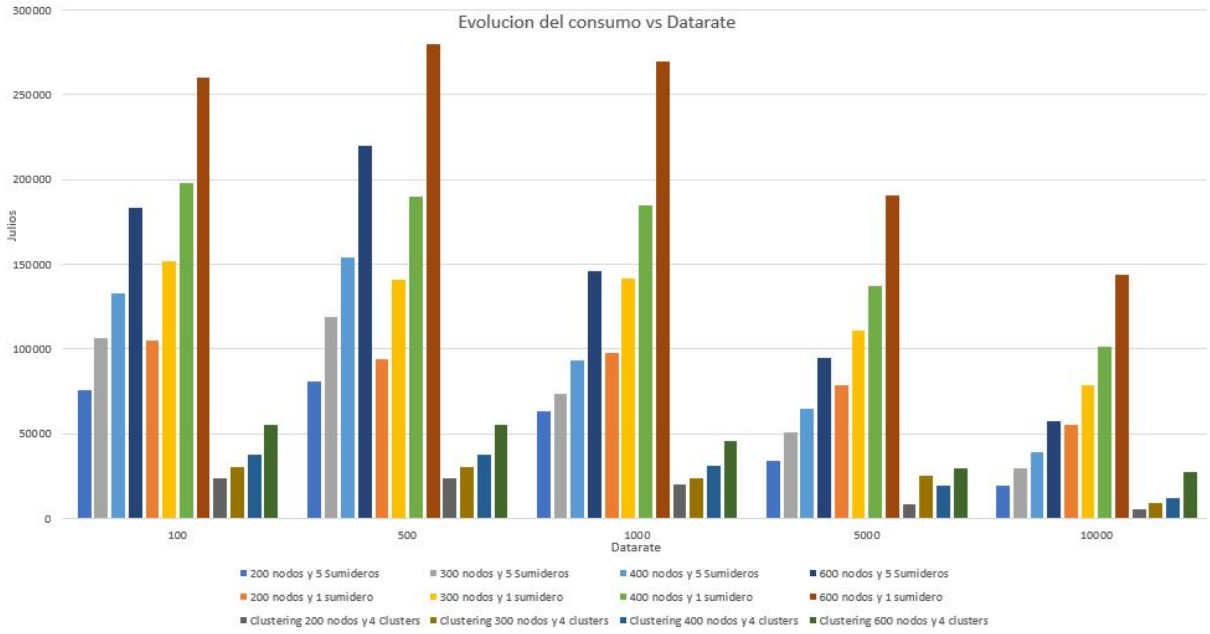

Figura 6.26: Comparación energética de las tres implementaciones

Una vez se ha visto que la implementación con el protocolo DBR con 5 sumideros obtiene mejores resultados a la hora de recibir paquetes, se tiene que ver de que manera los consigue. Para ello la figura 6.26 muestra una comparación total de las tres implementaciones, con diferentes valores de Datarate y con diferentes número de nodos. Aquí se ve claramente el beneficio de utilizar clustering: la potencia consumida independientemen-

te en que Datarate o con que cantidad de nodos es infinitamente menor. El resultado era obvio, ya que DBR utiliza siempre el 100 % de los nodos por su mecanismo de envío a base de Broadcasting (todos los destinos), mientras que en los clusters sólo están activos los nodos que envían (ya sean el nodo hoja origen o los líderes que reenvían su mensaje) nunca está la red completa transmitiendo. Estos resultados hacen que si comparamos el % de paquetes recibidos por el clustering (80) y su bajo consumo sin igual, hagan de él la alternativa más eficiente.

Como información complementaría en la figura 6.27 se obtiene el PDR con diferentes tamaños de paquete, donde sigue la línea de la figura 6.25, estando intermedio entre la implementación con DBR con 1 sumidero y 5 sumideros.

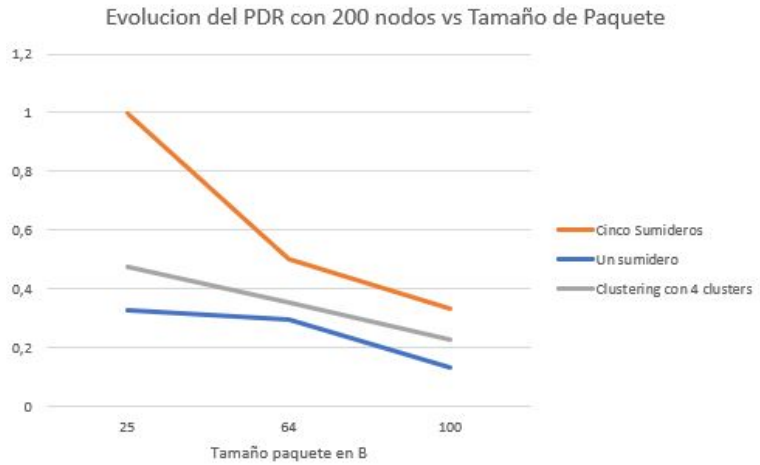

Figura 6.27: Comparación energética de las tres implementaciones

CAPÍTULO 7
Conclusiones y futuros Trabajos

El principal objetivo de este proyecto era la implementación y evaluación del protocolo DBR en base a [1] y [2] y posteriormente utilizándolo como base operativa para implementar sobre él un protocolo basado en el clustering. La eleccíon de éste protocola ha sido por su carencia de ningún tipo de algoritmo de configuración para las tablas de encaminamiento de los nodos, haciéndolo más moldeable y factible de evolucionar.

Se ha realizado un estudio para analizar el comportamiento del protocolo DBR con un sólo nodo sumidero, para ello se evalúa el Average energy tax (AET) VS número de nodos, donde se ha comprobado que realmente más consumo no significa más eficiencia en cuanto a paquetes recibidos con éxito. Siguiendo en esta línea, se ha valorado el PDR VS número de nodos, dónde se ha visto que contra más nodos tiene la red peor es la tasa de recepción. Todo ello se ha justificado al haber contrastado los resultados y explicar el cuello de botella que tiene la red.

En segunda instancia se ha analizado el protocolo DBR con 5 nodos sumideros. Para verificar su funcionamiento se ha experimentado con el parámetro Datarate, Delta, Umbral y Velocidad de los nodos. Con el primero de ellos se ha obtenido la conclusión de que aumentando el Datarate, se inyecta más rápidamente los paquetes a la red y por tanto hay menos probabilidad de que colisionen mientras un uno está en envío. Además si se transmite un paquete más grande, se tarda más en transmitir y se aumenta la probabilidad de colisión que previamente se había reducido con el Datarate. Del mismo modo se ha comprobado que contra más mayor Datarate, el tiempo que un nodo está transmitiendo y consumiendo es menor. También se ha cumplido que contra más nodos más consumo.

El parámetro Delta se ha analizado con valores de 25, 50 y 100 VS número de nodos de 200, 300, 400 y 600. Donde se ha comprobado que a medida que Delta aumenta, el consumo de energía también aumenta. Esto a su vez provoca en cadena que el retardo de la red disminuya porque se retransmiten más paquetes. El valor umbral se ha analizado con valores 20, 40 y 80 y con el mismo número de nodos que con Delta. En este caso si se aumenta el umbral disminuye el consumo porque se reduce el radio de nodos que son capaces de retransmitir, de igual manera, cuando esto ocurre el PDR se ve afectado negativamente, así como el retardo total si se aumenta el número de nodos.

Por último se ha estudiado la velocidad, con los valores de 1, 4 y 15 m/s y se ha llegado a la conclusión de que cuando hay más velocidad la estructura de la red se destruye antes y esto repercute negativamente en el PDR, así como en el consumo.

En tercera instancia se han comparado ambos resultados, con un nodo sumidero y con 5, dónde siempre se ha obtenido un mayor consumo y retardo por parte de la simulación con un nodo sumidero. Por contra el PDR siempre disminuye a cause del cuello de botella de la red.

Finalmente se han obtenido los resultados con la implementación con clusters. De ello se han derivado las conclusiones de que con un mayor número de cluster se obtiene un mayor consumo (hay más nodos líderes de cluster activos y transmitiendo) así como una tasa de recepción peor (dado que hay más probabilidad de que dos líderes envíen a un mismo destinatario).

Por último se han comparado los resultados de las tres implementaciones (DBR con 1 sumidero, DBR con 5 sumideros y clustering con 4 clusters). Se ha comprobado que el más eficiente sin duda es la versión con clusters dado que aunque no alcanza el 90 o 99 % de tasa de recepción del DBR de 5 sumideros, se queda en 80 % y con un consumo más de 3 veces menor.

Con todo ello, se han alcanzado los objetivos planteados el principio de implementar y valuar el protocolo DBR y a partir de él efectuar una alternativa con clusters más eficiente.

7.1 Relación con los estudios cursados

Al realizar este trabajo se han utilizado conocimientos adquiridos durante todos los años del Grado de Ingeniería Informática. La motivación proviene gracias al cursar asignaturas como Redes de computadores que sentó las bases para conocer los principales protocolos de encaminamiento en red, su funcionamiento o las redes inalámbricas.

Posteriormente también se obtuvieron conocimientos de la asignatura de Tecnología de sistemas de información en la red, en donde se dieron conocimientos para redes tolerantes a fallos o los principios del Spaning Tree, el cual se pretende imitar a la hora de configurar los Clusters y los nodos líderes entre sí.

También cabe destacar asignaturas de la rama de Ingeniería del Hardware como Configuración, Administración y gestión de redes o Tecnología de Redes que sentaron precedentes del proyecto.

Por último, y en cuanto a la parte práctica del mismo, se ha implementado con el lenguaje c, c++ y python en un entorno Ubuntu con manejo de la programación con Bash. Conocimientos que de adquirieron en asignaturas como Fundamentos de Sistemas Operativos, Diseño de Sistemas Operativos, Arquitecturas avanzadas o python en menor medida en Seguridad de Sistemas Informáticos.

7.2 Publicaciones

Como fruto de este trabajo se ha realizado una publicación en el XIII International Workshop on Sensors and Molecular Recongition, José Navarro, Izan Catalán, Alberto Bonastre, Rafael Ors, Juan Vicente Capella: “Desarrollo de un modelo de protocolo inteligente para redes de sensores subactuaticas”. XIII International Workshop on Sensors and Molecular Recognition (IWOSMOR XIII), València 2019.

7.3 Trabajos Futuros

Este trabajo final de Grado da lugar a una continuación para un Trabajo Final de Máster con el objetivo de evolucionar plenamente hacia un protocolo clusterizado, robusto y tolerante a fallos.

En la línea para conseguir eso, lo primero que se propone para mejorar es crear canales propios de los clusters, por lo que cada nodo tendrá internamente del cluster un canal para comunicarse con los nodos de su mismo cluster y otro canal externo para comunicarse con los demás nodos de la red (este sería el caso de los nodos líderes con los sumideros.

Además se ampliará la simulación para que los nodos dispongan de movilidad dentro del escenario. De este modo será más difícil mantenerse dentro de los clusters porque el cambio e posición hará que se acerquen o se distancien unos a otros, creando y destruyendo nuevos clusters. Para ello se implementaría un temporizador para que cada cierto tiempo comprobara si hay otro nodo líder más cerca. También el mecanismo tolerante a fallos deberá resistir perdidas y nodos caídos, así como organizar un plan para poner en hibernación ciertos nodos y activar otros en tiempo real para ahorrar consumo.

Por último se intentaría buscar alternativas para agrupar a los nodos dentro de un cluster, en lugar de por cercanía geográfica, podría ser por potencia de la señal. De igual manera, también se pretenderá utilizar técnicas inteligentes como Deep Learnig para programar el comportamiento de los nodos.

Bibliografía

[1] DBR: Depth-Based Routing for Underwater Sensor Networks , Hai Yan, Zhijie Jerry Shi, and Jun-Hong Cui. Department of Computer Science and Engineering University of Connecticut, Storrs, CT 06269-2155

[2] A Novel Cross-Layer Routing Protocol Based on Network Coding for Underwater Sensor Networks, Hao Wang, Shilian Wang, Renfei Bu and Eryang Zhang. College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410000, China. 2017

[3] Homepage of ns-3: http://www.nsnam.org/. [Accedido Septiembre de 2019]

[4] Diagonal and Vertical Routing Protocol for Underwater Wireless Sensor Network, Tariq Ali, Low Tang Jungb, Ibrahima Faye.

[5] Energy-efficiency and reliability in MAC and routing protocols for underwater wireless sensor network:Asurvey. Nusrat Zerin Zenia, MohammedAseeri, MuhammadR.Ahmed, Zamshed I.Chowdhury a, M.ShamimKaiser. Institute ofInformationTechnology,JahangirnagarUniversity,Savar,Dhaka,Bangladesh. 2016

[6] Routing protocols based on node mobility for Underwater Wireless Sensor Network (UWSN): A survey Mukhtiar Ahmeda, Mazleena Salleha, M.Ibrahim Channab. Department of Computer Science, Faculty of Computing, UTM, Malaysia, 2016.

[7] COMPARACIÓN DEL DESEMPEÑO DE LOS PROTOCOLOS DE ENRUTAMIENTO AODV Y DSR SOBRE UNA RED MANET EXPERIMENTAL, ESTEFANÍA ARAGÓN MONROY, UNIVERSIDAD MILITAR NUEVA GRANADA, Bogotá, 2012.

[8] Underwater Sensor Network Applications: A Comprehensive Survey, Emad Felemban,1 Faisal Karim Shaikh, UmairMujtaba Qureshi, Adil A. Sheikh, and Saad Bin Qaisar. Umm Al-Qura University, Mecca 21955, Saudi Arabia.

[9] Redes inalámbricas de sensores: una nueva arquitectura eficiente y robusta basada en jerarquía dinámica de grupos, Juan Vicente Capella Hernández. Editorial Universidad Politécnica de Valencia, 2010.

[10] A Survey on Underwater Wireless Sensor Networks: Progresses, Applications, and Challenges. J. Premalatha1 and Joe Prathap P M, Research Scholar, Dept.of CSE, Sathyabama University, Chennai, India.