ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

Gestión de Recursos en Redes Malladas
(Resources Management on Mesh Networks)

Para acceder al Titulo de

INGENIERO DE TELECOMUNICACIÓN

Autor: José María Acha Fernández

Julio - 2012

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: José María Acha Fernández
Director del PFC: Ramón Agüero Calvo
Título: "Gestión de Recursos en Redes Malladas"
Title: "Resources Management on Mesh Networks"

Presentado a examen el día: 26/07/2012

para acceder al Título de

INGENIERO DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Irastorza Teja, José Ángel
Secretario (Apellidos, Nombre): Agüero Calvo, Ramón
Diez Fernández, Luis Federico
Vocal (Apellidos, Nombre): Arce Diego, José Luis

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del PFC
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Proyecto Fin de Carrera Nº
(a asignar por Secretaría)

Índice

Índice de Figuras

Resumen Ejecutivo

En este documento se presenta un análisis sobre la influencia que tiene el uso de estrategias de gestión dinámica de recursos sobre las redes malladas. El innegable aumento de aparatos con conectividad inalámbrica hace totalmente necesaria la búsqueda de nuevos métodos para optimizar el uso de los recursos ya existentes. En este proyecto se intentan buscar mejoras tanto en el ámbito energético, como en el rendimiento. Para ello, se ha propuesto un modelo sencillo que se ha implementado sobre el *Network Simulator 2* y del que se han obtenido unos resultados que se han sometido a análisis.

Por último, para el despliegue de los escenarios se han tomado 4 estrategias diferentes y para medidas de rendimiento se ha usado un modelo no incluido en el simulador, pero de muy sencilla implementación.

Palabras Clave. Gestión Dinámica de Recursos, Network Simulator, Redes Malladas.

Abstract

In this document an analysis is showed about the influence of the dynamic resources management use in mesh networks. The undeniable increase of wireless connectivity devices do totally necessary the search of new methods to optimize the use of the already existing resources. In this project improvements it is tried to search improvements as in the energetic field, as in the throughput. For it, a simple model to set up on the *Network Smulator 2* is proposed and finally some results have been obtained and have been analyzed.

At the end, for the scene deployment, four different strategies have been taken to get these results. To measure the throughput a new model, which is not included in the simulator, has been used but his implementation is very easy.

Keywords. Dynamic Resources Management, Network Simulator, Mesh Networks.

Agradecimientos

En primer lugar me gustaría agradecer a mi tutor Ramón (Mon para los amigos), su paciencia y comprensión a la hora de ayudarme con la elaboración de este proyecto, se que ha sido un año duro para él tanto como para mí, pero siempre que ha podido ha estado ahí. Creo desde mi punto de vista es de los mejores profesores tanto a nivel académico como personal.

A continuación agradecer al grupo de trabajo en el laboratorio por ser tan abiertos a todo el mundo y siempre que lo he necesitado me han ayudado sin dudar. Sobre todo a Luisco, que creo que sin su ayuda este proyecto no podría haberse llevado a cabo, ya que aunque es ultra-exigente, siempre me ha apoyado con los medios de los que disponía.

A mis compañeros de clase que han hecho tan amenas esas clases que a veces son tan pesadas, en especial a mi grupo de aventuras (Chuchi, Antolín, Bello, Litos, Luis, Arnaldo, Marta, Pedrín y Arturo) con los que he pasado tantas y tantas horas en las bibliotecas de la Universidad de Cantabria y en la dura vida del ocio.

A mi círculo de amigos, que siempre están ahí que aunque la mayoría estudiemos en diferentes lugares siempre estamos en contacto y en cuanto podemos nos reunimos. Da gusto poder despejar la mente con semejante grupo. Creo que sin ellos no podría haber llegado a ser la persona que soy hoy.

Y por último a mi familia, sin duda el hombro en el que apoyarme cuando estoy a punto de desfallecer. Ellos han aguantado todo tipo de frustaciones y alegrías durante toda mi vida, y estoy muy contento de poder ofrecerles este momento tan especial para mí.

1 Intro

En este primer capítulo se hace un pequeño resumen de los motivos que han impulsado la realización de este proyecto. A su vez se da una visión general de los capítulos en los que se ha dividido el mismo. Finalmente se citan los objetivos principales.

1.1 Planteamiento del problema

Hoy en día a nadie le extraña ya el uso de dispositivos con conexión inalámbrica. Redes que siguen el estándar 802.11 del IEEE o, comúnmente llamadas, redes *WiFi*, están hoy al alcance de cualquier usuario.

Las tecnologías malladas, ofrecen la posibilidad de crear redes multi-camino y multi-salto entre los diferentes nodos que las constituyen. Gracias a esto, se pueden crear escenarios inalámbricos donde, sin necesidad de gran alcance de transmisión (sólo el necesario para llegar a otros dispositivos vecinos) se puede encaminar los datos hasta un punto deseado, por ejemplo un Gateway. Esto supone, entre otras cosas, un ahorro energético.

Pero, ¿es posible optimizar este ahorro?. Las técnicas de gestión dinámica de potencia que se tratan en este proyecto intentan abordar este tema, buscando una potencia de transmisión no constante, sino variable, dependiendo de la cercanía del próximo salto o identificando, incluso, si la ruta escogida es óptima a niveles energéticos, y si podría existir una alternativa mejor.

1.2 Objetivos

El objetivo primordial es comprobar las mejoras que aporta el uso de estrategias de gestión dinámica de recursos radio en despliegues con redes malladas, mediante el uso del

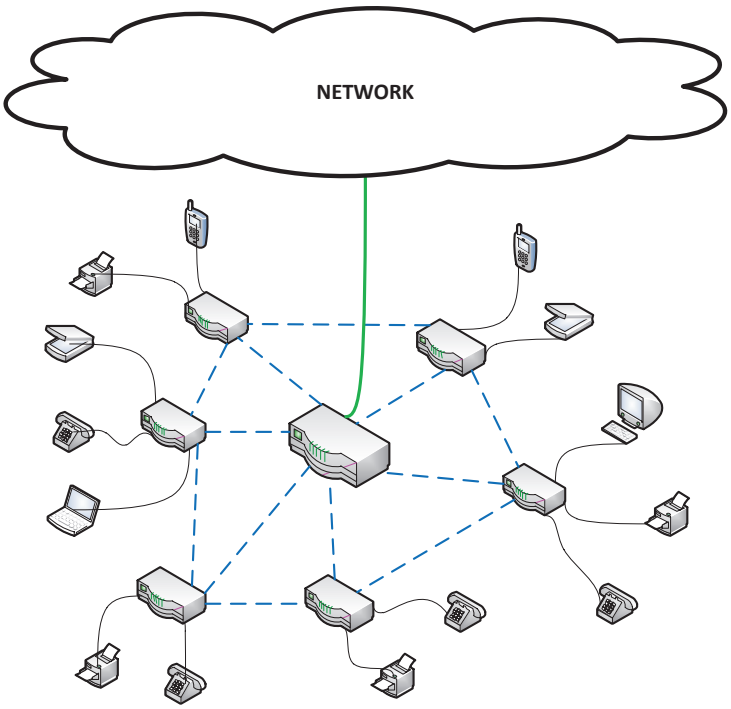


Figura 1.1: Ejemplo de red mallada

simulador *Network Simulator*.

Otros objetivos destacables serían el desarrollo e implementación, dentro de la estructura del simulador anteriormente citado, de todos los algoritmos necesarios para poder llevar a cabo estas tareas de gestión dinámica, así como la simulación y extracción de resultados.

Cabe mencionar que este análisis ha buscado comprobar la relación entre el número de saltos necesarios para alcanzar un Gateway desde cualquier nodo de la red y la potencia que se utiliza al insertar un flujo de tráfico en estos enlaces, añadiendo las citadas tareas de gestión. Para ello se ha dispuesto de cuatro estrategias de despliegue diferentes que se han sometido a análisis.

1.3 Estructura de la memoria

A continuación se detalla la estructura y contenido de los capítulos de esta memoria:

- En el Capítulo 2 se realiza un acercamiento al estado de la tecnología sobre gestión dinámica de potencia y de los recursos radio. Además se hace una descripción de los

mecanismos de despliegue utilizados y el algoritmo de encaminamiento usado. Se trata de unos métodos que ya están implementados o que se han usado con anterioridad, pero que han sido de gran ayuda para la consecución de los objetivos de este proyecto. Por último se describen los elementos externos al simulador NS2, no incluidos en las librerías iniciales del programa, sino que se han añadido posteriormente y que han sido necesarios en el desarrollo del trabajo.

- En el Capítulo 3 se presenta la herramienta de simulación utilizada durante todo el proyecto para crear escenarios acordes, para posteriormente analizar los resultados que se obtienen de ella. Se ha escogido *Network Simulator 2 (NS2)*. Se hará una breve presentación de la misma centrándola en un marco histórico; después se describirá su funcionamiento y, por último, se introducirán de manera breve los cambios realizados en la estructura interna del simulador para poder afrontar las fases del proyecto.

- El Capítulo 4 recoge todos los desarrollos, modificaciones y nuevas funciones que han sido necesarias para llegar a la consecución de este proyecto. Se empieza tratando las modificaciones realizadas sobre el protocolo de enrutamiento (*Static Routing*), necesarias para poder adaptarlo a la gestión de recursos. Después se describe con detalle la entidad *Monitor*, que proporciona conocimiento de todos los aspectos de la red que sean pertinentes para los objetivos del proyecto, así como las modificaciones al algoritmo de Dijkstra que se utiliza para la búsqueda de las rutas óptimas. Por último se explican los cambios en la clase *WirelessPhy*, que es la encargada de manejar la capa física, siendo muy útil para realizar cambios en la potencia utilizada.

- En el Capítulo 5 se recogen los resultados del trabajo realizado. En primer lugar se realiza un análisis del número medio de saltos en las rutas, después se analiza el gasto en potencia y, por último, se estudia la posibilidad de incorporar la calidad del enlace. Se hace uso de dos estrategias, una con gestión dinámica a nivel de paquete y otro mediante el uso del Dijkstra modificado. En este apartado se comparan resultados obtenidos, y debido a la cantidad de parámetros modificables y la cantidad de datos, se utiliza una representación gráfica para poder mostrar más claramente diferencias y similitudes entre las soluciones desarrolladas.

- El Capítulo 6, a la luz de los resultados presentados en el capítulo anterior, intenta obtener una serie de conclusiones sobre lo que aporta el uso de las técnicas de gestión de recursos, y se intentan proponer nuevas líneas de investigación que pueden mejorar lo existente y que han surgido durante el desarrollo del presente trabajo.

2

Estado del Arte

Este capítulo busca ilustrar al lector con los conocimientos necesarios para poder entender el marco en el que se ha desarrollado este proyecto y comprender así los asuntos tratados en el mismo.

2.1 Gestión Dinámica de Recursos Radio

Las redes malladas inalámbricas (*Wireless Mesh Networks, WMN*), representan un posible camino para la evolución de las comunicaciones inalámbricas, por la extensión que permite de las redes de único salto hacia las multi-salto. La WMN es una arquitectura de dos niveles, primeramente, los routers inalámbricos *Mesh Point (MP)* forman una columna multi-salto auto-organizada con uno o más *Internet Gateways (GWs)*; por su parte los *Mesh Access Points (MAPs)* proveen una infraestructura de acceso a los usuarios finales, formando una malla de nodos. Con esta idea se puede desplegar redes robustas, rápidas y flexibles; además de configurables sin necesidad de la costosa infraestructura cableada.

Las WMNs se enfrentan a numerosos retos: en primer lugar, la opción multi-salto, multi-canal, tiene que balancear la máxima conectividad (posible compartiendo sólo un canal) y la mínima interferencia entre transmisiones (usando diferentes canales). Por otro lado, casi todo el tráfico fluye a través del GW, resultando un cuello de botella, ya que el GW limita la capacidad de la red; la carga en el enlace decrece a medida que nos alejamos del Gateway o a medida que aumentamos el número de MAPS. Por último, presenta pobres propiedades de imparcialidad. Contención, retraso y pérdidas de paquetes incrementan con el número de saltos, algunos MPs más lejanos al GW están en desventaja frente a los más cercanos, quedando aislados cuando la carga de tráfico supera la capacidad de la red.

En [1] se propone un Gestor de Recursos Radio (RRM) que maneja eficientemente canales, potencia y velocidades en MPs multi-radio, para intentar paliar los problemas anteriores . Muchos RRM están coordinadas por una única entidad con conocimiento global

de la red. Respecto a la Asignación de Canales (CA) proponen un modelo híbrido, donde ciertos interfaces radio tienen canales fijos y otros cambian dinámicamente entre los restantes. Esto garantiza conectividad con todos los vecinos sin necesidad de cambios MAC. El Control de Potencia (PC) debe ser adaptado para ofrecer conectividad y minimizar la inteferencia y consumo energético. En cuanto a la Adaptación de Velocidad (RA), proponen una solución centralizada, equilibrando la velocidad seleccionada con la interferencia.

La contribución de [1] para la RRM consiste en construir una capa de abstracción, independiente de la capa radio subyacente en la operación de múltiples radios y transparente a las capas superiores. Ésta aplica las estrategias anteriormente mencionadas para alcanzar el máximo rendimiento en WMN.

2.2 Estrategia de Despliegue de Escenarios

En el anterior punto se ha intentado dar una idea del estado de la tecnología en materia de recursos radio y gestión de potencia. En el presente, se va a tratar el modelo seguido para crear los escenarios que posteriormente se analizaron en este proyecto. Estas estrategias ya se estudiaron en el documento [2].

En primer lugar, este modelo de gestión se puede definir como distribuido, en el que el primer objetivo es establecer el número de Gateways para garantizar una cobertura adecuada. Dado un despliegue sobre un escenario geométricamente limitado, que se supondrá siempre cuadrado, el número de gateways debe reflejar al número de subredes, o sub-grafos, que se forman, así como el radio de cobertura de los nodos teniendo en cuenta además un número máximo de saltos. Un primer parámetro que permite caracterizar la red es su grado de conectividad, que viene definido en la Ec. 2.1, donde N representa el número total de nodos de la red y SG el número de sub-grafos, entendiendo por sub-grafos los conjuntos de más de un nodo con conectividad entre sí; obviamente, la falta de conectividad vendrá dada por el número de nodos que no pertenecen a ningún sub-grafo y,que por tanto, se encuentran aislados.

$$\xi = \frac{N - SG}{N - 1} \tag{2.1}$$

El grado de conectividad tomará valor 1 en el caso en que de todos los nodos pertenezcan al mismo sub-grafo y 0 cuando todos los nodos se encuentren aislados.

Parece lógico pensar que la conectividad dependerá del radio de cobertura de los nodos que forman la topología de la red (R), de la superficie en la que ésta se despliega (cuadrada de lado L) y del número de nodos de la misma (N). De esta forma, se define la cobertura normalizada ρ como la relación entre R y L, y la densidad de nodos como la relación entre N y L^2 (Ec. 2.2).

$$\rho = \frac{R}{L}\ ;\ D = \frac{N}{L^2} \tag{2.2}$$

Se observa que, manteniendo contantes el número de nodos (N) y la cobertura normalizada (ρ), el producto de D por R^2 (Λ) no varía, como se muestra en la Ec. 2.3

$$\Lambda = D \cdot R^2 = \frac{N}{L^2}\rho^2 L^2 = N\rho^2 \tag{2.3}$$

Analizando el comportamiento de ξ frente a Λ se obtiene empíricamente una función que relaciona ambos parámetros y que se muestra en Ec. 2.4. La Fig.2.1 ilustra cómo varía la conectividad en función del número de nodos y de la cobertura normalizada.

$$\xi = f(\Lambda) = 1 - e^{-1,6\Lambda} = 1 - e^{-1,6N\rho^2} \tag{2.4}$$

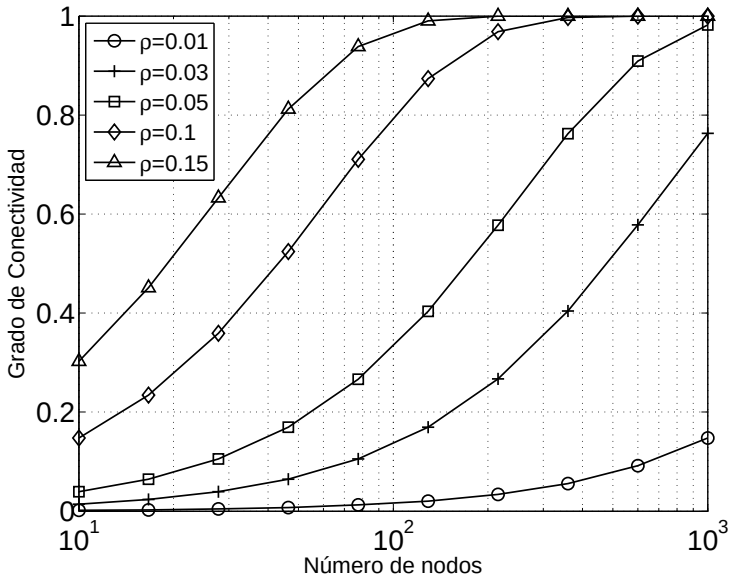


Figura 2.1: Grado de conectividad

Mediante un análisis fundamental, se ha estudiado, asimismo, la función densidad de probabilidad del número de sub-grafos que se crean, atendiendo a diferentes topologías de red. En la Fig. 2.2 se muestra el número de sub-grafos esperados para dos topologías con igual número de nodos y diferente conectividad; es decir, dos topologías en las que se ha variado la cobertura normalizada ρ. A partir de dicho comportamiento se puede determinar el número de gateways necesarios, en cada caso, para alcanzar una cobertura determinada.

Una vez analizadas las características de un despliegue aleatorio de nodos, es necesario distribuir de una manera adecuada el papel de gateway entre todos los nodos que conforman

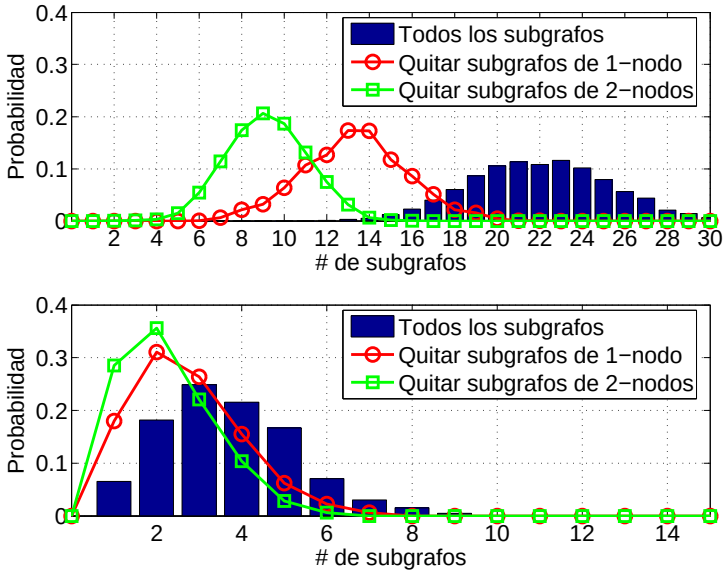


Figura 2.2: Función densidad de probabilidad de subgrafos para una red de 80 nodos sobre dos escenarios con diferentes grado de conectividad (figura superior, $\xi = 0{,}7$; figura inferior, $\xi = 0{,}9$)

la red. Este reparto dependerá de la probabilidad de cobertura, del adecuado reparto de los nodos entre los gateways y de la sobrecarga de tráfico en la red o, adicionalmente, para minimizar el número de saltos necesarios para llegar al gateway, parámetro íntimamente ligado con la eficiencia energética.

Las estrategias que se exponen a continuación han sido desarrolladas y estudiadas en [2].

2.2.1. Estrategia 1: despliegue aleatorio

En este caso el papel de gateways se asigna de forma completamente aleatoria entre los nodos que componen la red, ignorando la topología creada. Esta estrategia refleja la situación más desfavorable ya que, debido al carácter aleatorio de la asignación, pueden darse situaciones en que ciertos gateways permanezcan aislados del resto de nodos. Esta estrategia permitirá comparar la prestaciones del resto con un peor escenario.

2.2.2. Estrategia 2: despliegue óptimo geométrico

En este caso también se ignora la topología de la red a la que ha dado lugar el despliegue de nodos, pero se asume que los gateways se sitúan en aquellos puntos que

aseguran una mayor cobertura (*geométrica*) del área bajo análisis.

Esta estrategia no lleva a cabo, propiamente dicho, una asignación de papeles entre los nodos desplegados, si no que, una vez desplegado el escenario, desplaza tantos nodos como gateways se deban desplegar a los lugares adecuados y les asigna, posteriormente, el papel de gateway.

2.2.3. Estrategia 3: despliegue óptimo topológico

Partiendo del conocimiento total de la topología de la red, se asigna el rol de gateways a aquellos nodos que garanticen un *coste* global menor, entendiendo como coste el número total de saltos necesarios para alcanzar a los gateways. Para resolver este problema se utiliza la *p-mediana* [3], que establece un conjunto de M gateways que minimicen la siguiente función:

$$\sum_{j<N} \left\{ \min_{i<M} d_{ij} \right\} \tag{2.5}$$

en la que N es el conjunto de nodos y $\min d_{ij}$ es la distancia mínima, en número de saltos, obtenida mediante el algoritmo de Dijkstra, entre los nodos i y j.

Uno de los problemas que se le achacan a la *p-mediana* es la necesidad de dar servicio a todos los nodos que componen la red (esto es, la demanda ha de ser satisfecha), ya que, de lo contrario, el problema de la *p-mediana* no tiene solución; así, en caso de no disponer de suficientes gateways para cubrir toda la red, el simulador propietario es capaz de eliminar, de forma iterativa, los sub-grafos de menor tamaño hasta que el algoritmo encuentra una solución al problema.

2.2.4. Estrategia 4: despliegue sub-óptimo topológico

Dado el inconveniente que supone la necesidad de dar servicio a la totalidad de los nodos en la resolución de la *p-mediana*, en función de las características particulares de la topología de la red, se podrían dar casos en los que fuera más interesante renunciar a la cobertura de ciertos nodos, con el fin de potenciar una distribución más ecuánime del resto de los nodos y, así, poder obtener un menor número de saltos para alcanzar a un gateway (Ec. 2.5).

En este sentido se propone una sencilla modificación al algoritmo tradicional de la *p-mediana*, en el que no se consideren aquellos sub-grafos con un tamaño inferior a ν nodos, siendo ν un parámetro de diseño, que tendrá que tener en cuenta el beneficio adicional

que supone no gestionar a un conjunto de nodos y la pérdida (en probabilidad de ser gestionado) correspondiente.

Esto supone que, a diferencia de la estrategia 3, se descartan, de forma preventiva, los sub-grafos de tamaño ν antes de la resolución del problema de la *p-mediana*; en caso de no obtenerse solución, debido al bajo número de gateways, se procederá de la misma forma que en el caso anterior.

2.3 Algoritmo de Encaminamiento: Dijkstra

El siguiente paso, tras el despliegue y elección de nodos y gateways, es proporcionar de la información necesaria a cada elemento para conocer el camino óptimo hasta el gateway más cercano. Para ello el método seleccionado ha sido el algoritmo de Dijkstra, del que se va a hablar en este apartado. Durante la realización del proyecto se han hecho ciertas modificaciones al esquema inicial del método pero estas serán comentadas más detalladamente en el Capítulo 4.

También denominado algoritmo de caminos mínimos, determina el camino más corto desde un vértice (nodo) de un grafo al resto de vértices, dando pesos a las aristas (caminos). Tiene el nombre de su creador *Edsger Dijkstra*, que lo escribió por primera vez en 1959.

El algoritmo podría resumirse en los siguientes pasos:

1. Se parte de la idea de que se tiene un grafo con N nodos, en el que se quiere conocer la distancia de X al resto de los nodos, para guardar estas distancias se inicializa un vector D con tamaño N.
2. Todas las distancias del vector D son puestas a un valor infinito relativo, ya que al principio son desconocidas, excepto la del propio nodo X, que se establece a 0, ya que la distancia de X a X es 0.
3. Se denomina a al nodo actual en el análisis, en la primera iteración será el nodo del que se pretenden conocer los caminos mínimos, es decir X.
4. Se recorren todos los nodos adyacentes de a, v_i
5. Si la distancia desde X hasta v_i guardada en D es mayor que la distancia desde X hasta a sumada a la distancia desde a hasta v_i, la distancia en D_i se sustituye por ésta última.
$$if(D_i > D_a + d(a, v_i)) \rightarrow D_i = D_a + d(a, v_i)) \tag{2.6}$$
6. Se marca el nodo a.

7. Se toma como nodo actual a aquel con menor valor de D, y se vuelve al paso 4 mientras existan nodos no marcados.

Una vez completado el algoritmo, el vector D contendrá las distancias mínimas a todos los nodos del grafo.

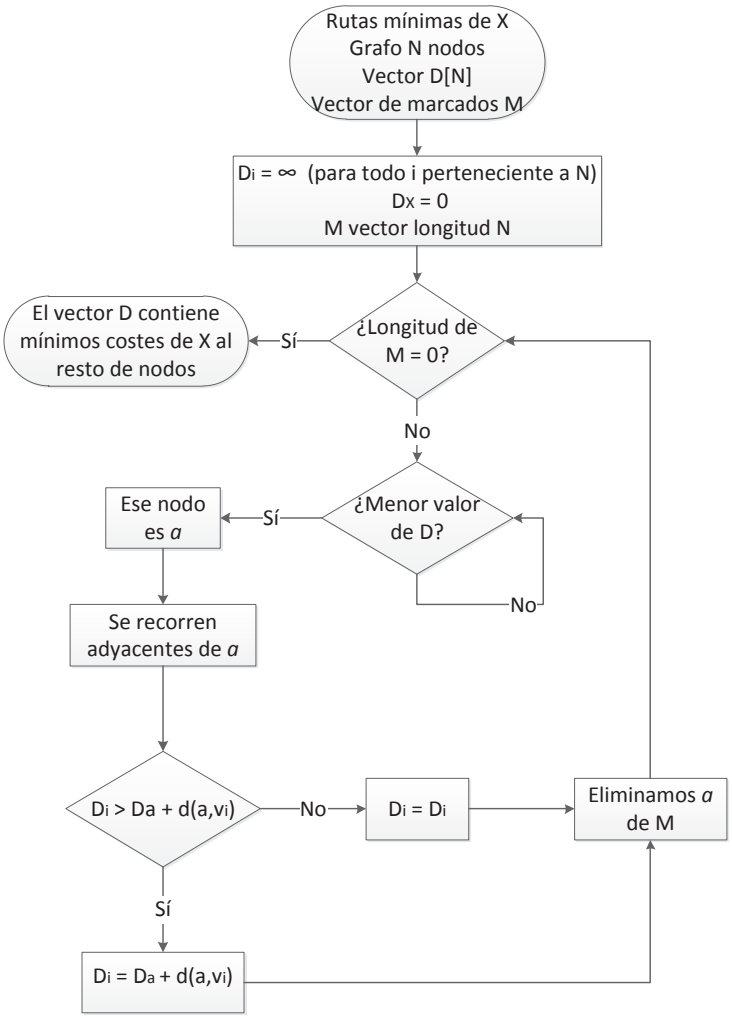


Figura 2.3: Diagrama de flujo del algoritmo de Dijkstra

2.4 Elementos Externos a NS-2

2.4.1. Modelo de propagación Simple Model

Uno de los problemas que ofrece NS2 es su escasa fiabilidad en los modelos de propagación de medio inalámbrico. Por ello, y para satisfacer las necesidades de este proyecto, se ha usado un modelo de canal que aun no siendo muy riguroso, resulta razonable y útil, para el uso que se le va a dar.

Los modelos que acompañan al simulador son: Free Space (regido por la *Fórumla de Friis*), Two-Ray Ground (asume además del camino directo, uno reflejado en el plano de Tierra) y, por último, "Shadowing"(añade componente aleatoria para representar los desvanecimientos provocados por propagación multicamino). Estos métodos sólo tienen en cuenta el cálculo de la potencia que llega al nodo y aplican las pérdidas a la potencia con la que se transmite la trama, sin considerar potencias de ruido, ni relaciones señal a ruido.

El nuevo modelo utiliza la probabilidad de pérdida de una trama al ser transmitida, la que llamaremos desde ahora *FER* (*Frame Error Rate*).

La configuración del canal Simple Model se lleva a cabo mediante la modificación de 3 parámetros fundamentales:

- Distancia máxima (dmax): mayor separación entre los nodos que realizan la comunicación, para distancias mayores los paquetes siempre se pierden.
- Parte de alcance sin pérdidas (α): Valor comprendido entre 0 y 1, indica el porcentaje de la distancia máxima donde no habrá perdidas. A partir de ese punto, éstas irán aumentando dependiendo del valor de *b*.
- Tendencia en el incremento de las pérdidas (b): mayor o menor hostilidad del canal en relación con la distancia. En el caso de *b=1* pérdidas y distancia tienen una relación lineal.

Una representación gráfica de la actuación del canal, se muestra en la Figura 2.4.

2.4.2. Static Routing

Una de las claves para poder abordar con éxito este proyecto, es que los paquetes se encaminen por las rutas ideadas teóricamente, mediante el algoritmo de Dijkstra, implementado en una entidad *Monitor* de la que se hablará en el Capítulo 4. El problema es que los protocolos planteados en NS2 no facilitan esta tarea, ya que su uso podría modificar

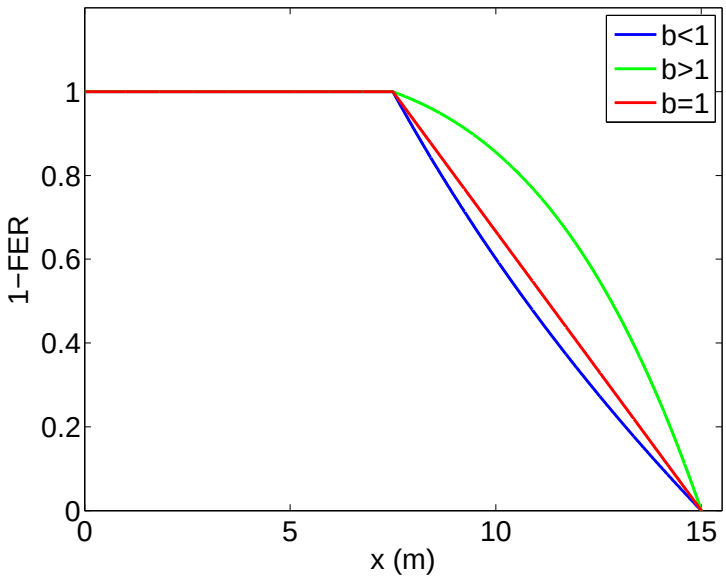


Figura 2.4: Simple Model - FER vs. Distancia

las rutas que se habían elegido primeramente,(usando procedimientos de descubrimiento), invalidando los resultados. Por ello, el modelo buscado debe facilitar la incorporación de rutas fácilmente configurables y que se mantengan fijas durante toda la simulación, para lo que se usa *Static Routing*. Además, en este trabajo se ha usado la clase *Static Routing* como enlace entre la capa física de NS2 recogida en el objeto *WirelessPhy* (del que se hablará más adelante) y el *Monitor*, sin necesidad de tener una conexión directa entre éste último y aquel.

Este protocolo dispone de varios módulos, pero en este proyecto sólo se usan los necesarios para establecer rutas simples, tal y como se describe brevemente a continuación.

- *Static Routing Table*: Como su nombre indica, es una tabla que recoge la información de todas las rutas desde un nodo a los demás. Cada entrada contiene el destino final, próximo salto de la ruta y, en el supuesto de usar más de una interfaz, el identificador de la correspondiente.
- *Static Routing Agent*: Hereda de la clase Agent de NS2 y gestiona y crea las tablas de rutas de cada uno de los nodos. Cada vez que un paquete se transmite a través de uno de los nodos, el objeto *Classifier* lo envía a este agente de encaminamiento para que compruebe el destino final y próximo salto.

3

Network Simulator

Este capítulo trata de ilustrar al lector sobre la herramienta de simulación utilizada durante todo el trabajo. Se abordarán, con más profundidad, los módulos que han sido de mayor utilidad durante la realización del proyecto. Además se terminará haciendo una aproximación las modificaciones realizadas en estos módulos, que se abordarán con mayor detalle en el Capítulo 4. Se ha tomado como referencia los documentos [4] y [5].

3.1 Introducción

El estudio de redes inalámbricas se ha convertido, dado a la gran variedad de usos que tiene en la vida real, en una de la tareas imprescindibles para su desarrollo e implementación. Debido a los altos costes y dificultad de obtención de resultados empíricos sobre entornos reales, el uso de herramientas de simulación supone un punto clave en la evolución de estos sistemas.

El simulador NS (Network Simulator), vio la luz en 1989 en el *Lawrence Berkeley National Laboratory* (LBNL) y actualmente está enmarcado en el proyecto *Virtual Inter-Network Testbed* (VINT) de la Universidad de California del Sur. Ha ido evolucionando gracias a instituciones como SAMAN, CONSER, ICIR, Sun Microsystems, la UCB Daedelus o Carnegie Mellon, que fueron los desarrolladores del código para redes inalámbricas. Se trata de un simulador de redes dirigido por eventos discretos, que actualmente se encuentra en su version 3, aunque en este proyecto se usará la 2.34. Se trata de una herramienta de código abierto, lo que facilita su uso en diferentes sectores: educación, sector empresarial, investigación o incluso Gobierno. Su estructura en módulos brinda una gran flexibilidad para poder agregar nuevos módulos externos con nuevas funcionalidades, aunque esto conlleva un conocimiento exhaustivo de su arquitectura interna.

3.1.1. Dualidad NS2

Una característica propia de NS2 es el uso de dos lenguajes de programación. Estos son *C++*, que implementa los componentes del simulador (agentes, nodos, enlaces...) mientras que *oTcl* se encarga de los escenarios de simulación.

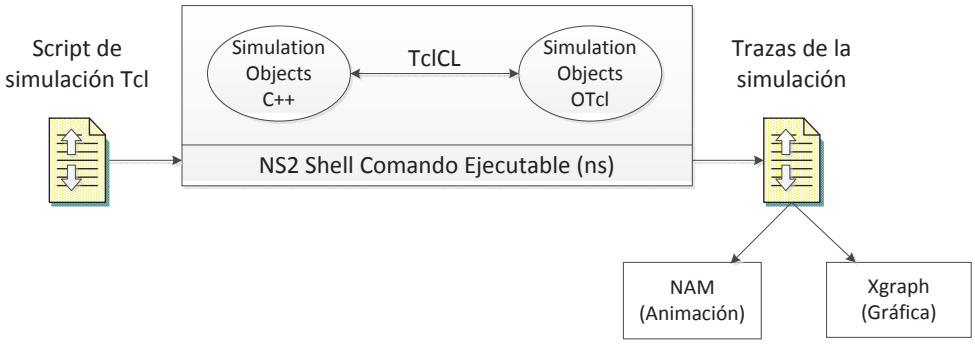


Figura 3.1: Estructura básica de NS2

La definición de los componentes de red, requiere de un lenguaje complejo, capaz de modelar los protocolos, modelos de canal, etc. En cambio, los escenarios y definición de parámetros se beneficiarían de una herramienta más simple y rápida. Para ello, C++ se encarga de la parte compleja, es decir el modelado de componentes de la red. C++ es un lenguaje orientado a objetos con gran capacidad jerárquica y nivel de estructuración, su inconveniente radica en que cada cambio requiere una recompilación de todo el código, lo que no lo hace idóneo para describir los escenarios a analizar, donde la modificación de parámetros durante la fase de realización de simulaciones es algo frecuente. En este momento es cuando entra en juego el segundo lenguaje, oTcl (*Object-Oriented Tool Command Language*), modificación orientada a objetos del Tcl, que permite la modificación de las especificaciones de componentes de la red y, al tratarse de un lenguaje interpretado, no necesita compilación.

Con esta dualidad se consigue que el usuario pueda abstraerse del funcionamiento de la red y sus componentes y sólo se concentre en la descripción de los escenarios. Ambos lenguajes tienen como enlace la librería *tclcl*. El esquema mencionado se muestra en la Figura 3.1.

3.2 Mobile Node NS2

Debido al carácter inalámbrico de todos los elementos relacionados con el proyecto, la creación de nodos del tipo *Mobile Node* parece algo casi intuitivo dentro del desarrollo de

simulación de NS2. En la Figura 3.2 se especifican las partes de las que consta este objeto, que se detalla a continuación, empezando por el Canal y hacia arriba.

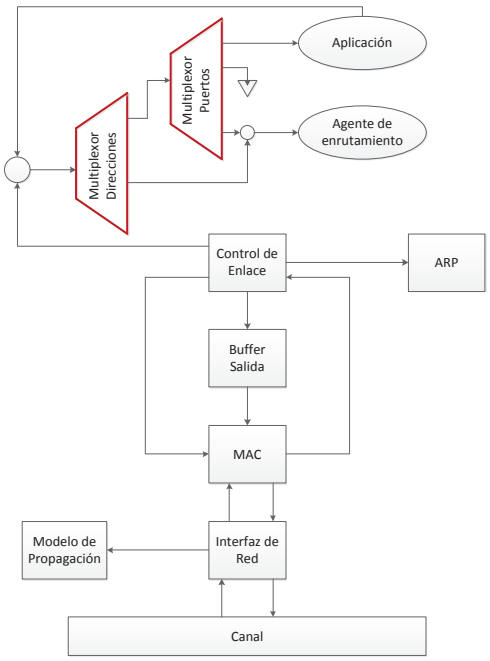


Figura 3.2: Estructura Mobile Node en NS2

- **Canal**. Representa el medio compartido por todos los nodos, que habilita la conexión entre los diferentes interfaces de red.
- **Interfaz de red (PHY)**. Sirve para emular el interfaz hardware, se encarga de acceder al medio físico, y controla los parámetros de potencia de transmisión, energía, tipos de antena... .
- **Modelo de Propagación**. A partir de la potencia de transmisión, disposición geométrica, y otras características del escenario, determina la potencia con la que será recibida las tramas en el destino. En este proyecto se ha usado un modelo sencillo, Two-Ray Ground, para los cálculos de potencia y, para la calidad del canal, el ya mencionado Simple Model.
- **MAC**. Implementa el protocollo MAC utilizado, principalmente el estándar 802.11. Permite simular colisiones, detecciones de portadora y genera, envía y recibe paquetes CTS, RTS, AC y DATA, tanto unicast como broadcast.
- **Buffer de Salida**. Mantiene los paquetes almacenados, hasta que estén listos para enviarse por el medio radio. A través de la clase *PriQueue* se otorga prioridad a los paquetes.

- **Control del Enlace**. Conecta el agente de enrutamiento con la tecnología de red subyacente. Tiene asociado un mecanismo de resolución de direcciones (ARP).
- **ARP**. Se encarga de añadir al paquete la dirección MAC del nodo destino. Si no tuviera esta dirección el paquete se almacena mientras se envía un paquete broadcast para conseguir dicha dirección.
- **Multiplexor de Direcciones**. Cuando llega un paquete y ha pasado el filtro MAC, este elemento lo procesa, en función de la dirección IP del paquete, y lo envía al segundo multiplexor (si el paquete está dirigido a ese nodo) o al protocolo de enrutamiento para ser retransmitido hacia su próximo salto.
- **Multiplexor de Puertos**. Una vez comprobado que el paquete se dirigía al propio nodo, se encamina al puerto de destino mediante esta entidad, que lo entrega al agente correspondiente.
- **Agente de Encaminamiento**. Implementa el protocolo de encaminamiento, por lo que mantiene una lista acerca de las rutas para alcanzar todos los destinos. Además gestiona el envío de paquetes de control.

3.2.1. Mobile Node de Múltiples Interfaces

Uno de los primeros cambios realizados en el simulador permite que el *Mobile Node* pueda funcionar en el caso de disponer de múltiples interfaces, en el documento [6] se detalla el procedimiento para conseguir la estructura de la Figura 3.3.

A continuación, se detallan los cambios más importantes que ha sufrido el simulador para llegar a esta estructura:

- Cambios en TCL
    1. Se añaden funciones a la librería de TCL. Estas librerías se pueden leer desde el script del escenario principal, y sirven para indicar el número de interfaces de cada nodo, y algunas variables ya existentes se convierten en arrays para albergar estas nuevas interfaces.
    2. La función que genera el nodo inalámbrico pasa de tener una única interfaz a varias, por lo que sus componentes (capa de enlace, cola de interfaz, capa física, interfaz con su correspondiente ARP y antena) se replican N veces. También se conecta cada canal con su interfaz asociado. Una limitación que aparece es que todas las salidas al canal tendrán las mismas propiedades de capa enlace y física, tipo de cola de interfaz, modelo de propagación y antena.

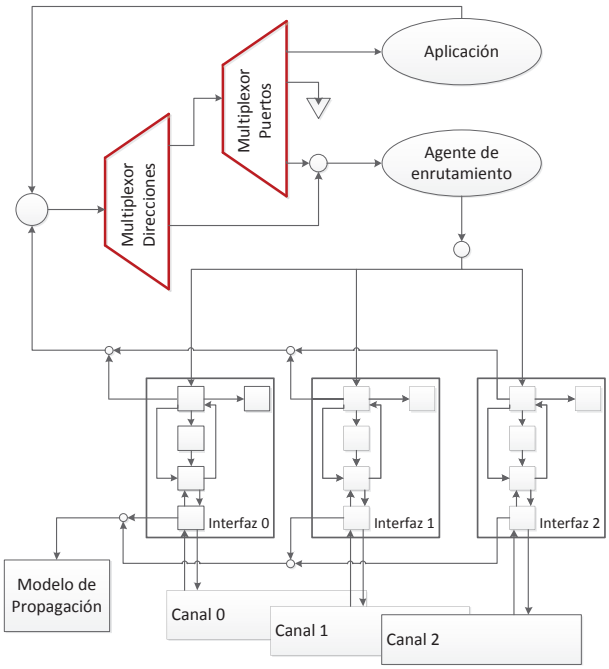


Figura 3.3: Estructura Mobile Node en NS2 ajustado a múltiple interfaz

3. Cuando se crea el nodo, también se crea el agente de enrutamiento, por lo que este debe saber que está gestionando múltiples interfaces. Lógicamente, el agente tiene que cambiar internamente, para que pueda recibir esta información y usarla adecuadamente.

- Cambios C++

1. El simulador controla los nodos pertenecientes a un mismo canal mediante una lista enlazada, que deberá extenderse para todos los canales que haya. Además, debe saber hacia que interfaz mandar los mensajes, dependiendo del canal del que provengan.
2. Acorde con lo dicho antes, hay que modificar las capas físicas y MAC para que cuando reciban un paquete, marquen en la cabecera la dirección de la interfaz por la que fue recibido.

- Cambios en el agente de enrutamiento

1. En el artículo [6] se expone un ejemplo con el caso de AODV. En este PFC no es necesaria esta parte porque se ha usado el *Static Routing* comentando en la Sección 2.4.2, aunque no sea estrictamente un protocolo de enrutamiento, ya que no usa mensajes de control.

4

Desarrollos en el marco de NS-2

Como ya se ha comentado en anteriores capítulos, para la realización de este trabajo ha sido necesaria la modificación e incorporación de nuevas funciones dentro del Network Simulator 2.

En este capítulo se detallan todos estos nuevos procesos y modificaciones , evitando el análisis profundo de comandos, funciones de C++, etc. Que harían la lectura mucho más compleja y árida.

4.1 Modificaciones en Static Routing

Las características básicas de este protocolo, que será utilizado a lo largo de todo el proyecto, ya han sido descritas en la sección 2.4.2, pero para poder atacar el tema de gestión de recursos radio, han sido necesarias ciertas modificaciones adicionales en su estructura.

Primeramente, comentar que los cambios de potencia, como ya se dijo en la sección 3.2, se realizan en la capa física del objeto *Mobile Node*, a través de una clase llamada *WirelessPhy*. Por lo que parece lógico que el primer paso fuera tener acceso desde el agente de enrutamiento a esta capa, para poder ejecutar ciertas ordenes. Hay que tener en cuenta que este acceso tiene que poder ser viable para el caso de una o varias interfaces.

Tras esto, NS2 no permitía modificar la potencia dinámicamente durante la simulación, ni siquiera se podía cambiar la potencia a cada nodo; el simulador, en el script de escenarios, permite modificar una potencia global, que será la que usen todos los nodos a la hora de transmitir. Esto no es deseable para los objetivos del trabajo. Otra vez a través del agente de enrutamiento, y recibiendo como parámetro la potencia nueva deseada, se ejecutó un primer paso, que consistía en modificar la potencia, siendo diferente para cada nodo o, incluso, para cada interfaz. El siguiente paso fue más sencillo, ya que poder modificar la potencia durante el tiempo de ejecución sólo necesitaba utilizar un comando ya existente en el escenario .tcl. En este proceso, fue necesario modificar ciertos aspectos en

el fichero *WirelessPhy*, como se detalla posteriormente en el apartado 4.3.

Por último, mas que hacer un cambio de potencia, directamente, resulta más útil conocer la distancia al próximo nodo de la ruta y, mediante esta variable, calcular la potencia necesaria con la que transmitir. Esto es lo que se ha implementado en una nueva función en la que el parámetro de entrada es la distancia, ya que calcula la potencia que es necesaria en la transmisión para enviarla a la capa física.

4.2 Power Monitor

Se puede decir que es la pieza clave de este trabajo. Hereda de la clase *TclClass*, con lo que tiene un nexo directo con los escenarios, esto lo hace muy útil ya que no requiere recompilación para que el propio usuario pueda usar sus funciones y variables. Es un elemento omnisciente, por lo que en cada simulación se creará un único Monitor, que contendrá toda la información necesaria acerca de nodos, rutas... Por último cabe destacar que, de ahora en adelante, se usará el caso de una única interfaz, para facilitar los análisis llevados a cabo; además, debido a que se ha usado el algoritmo de Dijkstra, el uso de múltiples interfaces no cumple una de las propiedades fundamentales de los pesos asignados a los enlaces llamada *isotonicidad*. En cualquier caso se dejan iniciados todos los aspectos necesarios para que más adelante se pueda integrar en el trabajo multi-interfaz.

A partir de ahora se van a detallar los procesos más importantes que tengan relación con los objetivos de este proyecto, ya que procesos como el registro de cada nodo con sus características en el monitor, o como se buscan nodos intermedios en una ruta, no aportan valor añadido frente a los objetivos de este trabajo.

4.2.1. Modelo de Coronas

El primer paso fue fijar un rango de cobertura máximo para los nodos. Este rango va a ser de ahora en adelante 15 metros, aunque es perfectamente variable, como ya se comentará en la sección 4.2.4. Sin embargo, esto no es eficiente, si se piensa en el caso de que un nodo este cerca del otro ya que, con el modelo actual se está transmitiendo a una potencia muy superior a la requerida y por lo tanto se estará consumiendo batería de manera innecesaria.

Como se muestra en la Figura 4.1 la distancia para alcanzar el nodo 1 (d1), es menor que la necesaria para alcanzar el nodo 2 (d2), por lo que obviamente sería necesaria menos potencia para transmitir al nodo 1 que la requerida para transmitir al nodo 2. Aún así, con el modelo actual se transmitiría a una potencia todavía mayor que sería la capaz de abarcar todo el rango de cobertura y, por lo tanto, se hace un uso muy ineficiente de la

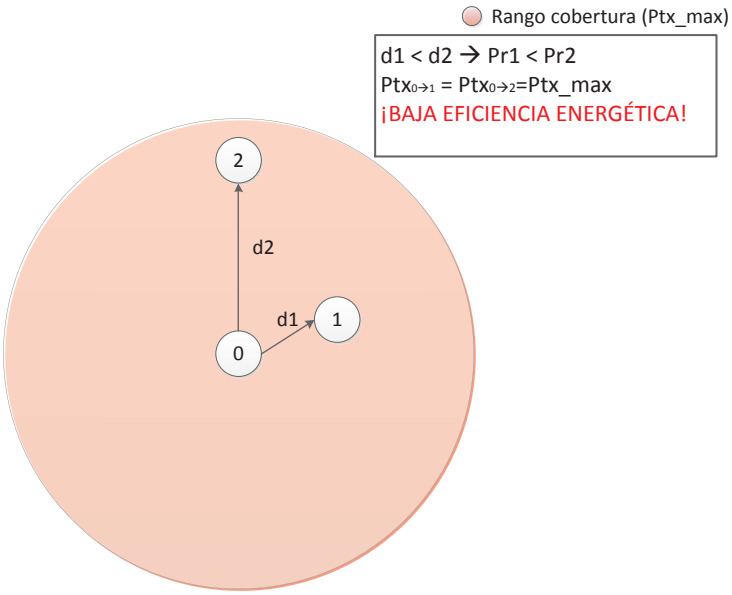


Figura 4.1: Modelo Antiguo de Cobertura

energía.

En este trabajo se propone un nuevo modelo, en el que el rango de cobertura se divide en círculos concéntricos a los que se denominarán coronas. Sobre cada corona se transmitirá una potencia diferente siendo mayor cuanto más lejos se esté del nodo origen. Con esto lo que se consigue es una mayor variedad de potencias de transmisión, mejorando el uso de la energía.

Como se ve en la Figura 4.2, el nodo puede transmitir a 3 potencias diferentes, por lo que, dependiendo de dónde se encuentre el nodo al que quiera transmitir, se usará una de estas tres potencias, por ejemplo, al nodo 2, como se situa en la corona 2 se transmite con potencia P_{tx2}, sin necesidad de tener que usar la máxima potencia como en el modelo tradicional.

Como es lógico pensar, cuanto mayor profundidad de coronas, es decir; cuanto mayor número de coronas se definan, serán necesarias más potencias posibles y, por lo tanto, se conseguirá una mayor eficiencia en términos energéticos. Pero como se verá más adelante esto no siempre tiene porqué ser beneficioso, ya que puede empeorar la calidad del enlace.

Como ya se comentó en la sección 2.3, se ha utilizado en este proyecto el algoritmo de Dijkstra para la búsqueda de camino de coste mínimo pero, ¿cómo asignamos el coste a las aristas?. Este problema se resuelve también mediante el uso del método de coronas. Dependiendo de en qué corona esté situado el nodo con el que se pretende conocer el peso, se asignan valores diferentes. Así, las rutas que surgirán usarán más saltos, pero en las transmisiones se usará menos potencia que en el caso de que sólo hubiese un salto.

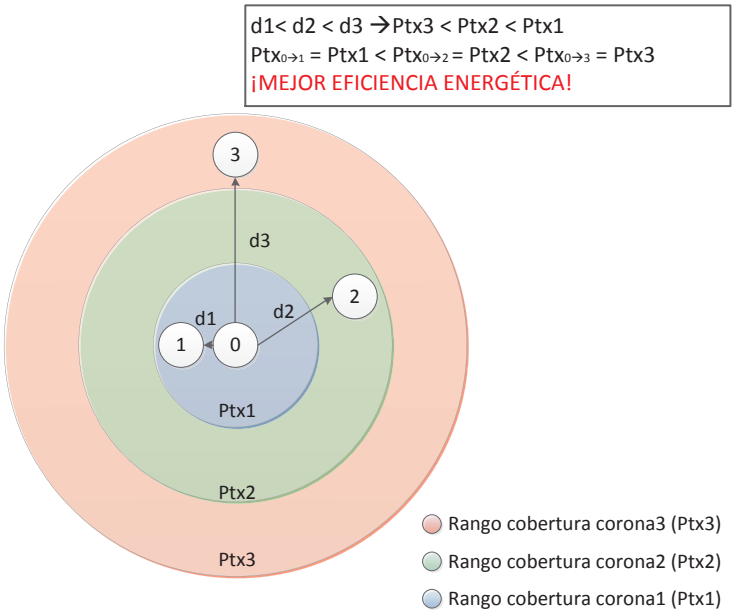


Figura 4.2: Modelo de Coronas

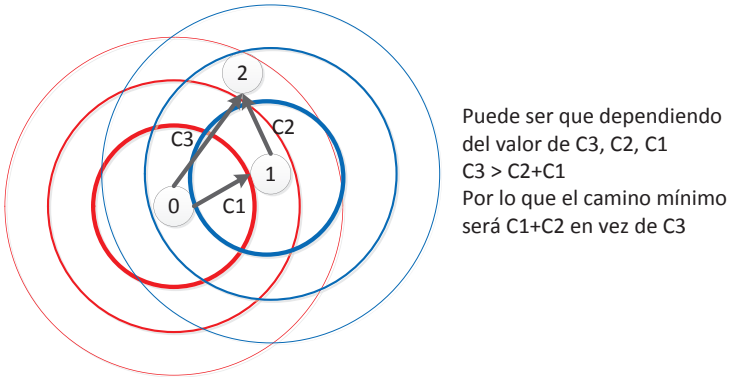


Figura 4.3: Modelo de Costes con las coronas

En la Figura 4.3 se ilustra el método de costes, viéndose que los círculos concéntricos muestran las coronas y rangos de los nodos 0 (rojo) y 1 (azul), por lo que dependiendo del coste que se asigne a cada corona, la ruta óptima podría ser el paso por 1 en vez del enlace directo a 2. Por ejemplo, con costes 1,3,5 para las coronas 1,2,3, respectivamente, se demuestra que el camino de 0 a 2 directo tendría un coste 5 que es mayor que el de 0 a 2 pasando por 1, que tendría un coste $1 + 3 = 4$.

La elección de los pesos que se le otorga a la corona es muy importante a la hora de elegir el modo a priorizar, si el de mayor saltos o el de menor; ya que para el mismo caso de la Figura 4.3 si los pesos fuesen 1,2,3 el camino óptimo seguiría siendo el directo, ya que a igualdad de pesos se escogerá el camino de menor saltos.

4.2.2. Elección de Gateways

Para la obtención de resultados que se comentarán en el capítulo 5, se ha usado un método de inyección de tráfico unidireccional desde un nodo a un gateway al que está conectado. Por lo tanto hay que idear un método de elección del mejor gateway, ya que en los escenarios que se estudiarán existe más de un gateway y el nodo sólo puede establecer conexión con uno.

Primeramente, el monitor debe tener el conocimiento necesario para identificar qué es gateway y qué no lo es. Para ello, se realiza el despliegue de todos los nodos y, posteriormente se le asigna el rol de gateway a un número de ellos. Esta información es la que conocerá el monitor.

Para establecer la conexión con el gateway correcto, se ha usado también la elección del de menor coste. Por lo tanto, y sin tener en cuenta gateways o no, se ejecuta el Dijkstra para el nodo que se pretende conectar a su mejor gateway. Una vez obtenidas todas las rutas a los demás nodos, se revisa la tabla en busca de los gateways, seleccionando aquel con el menor coste asociado.

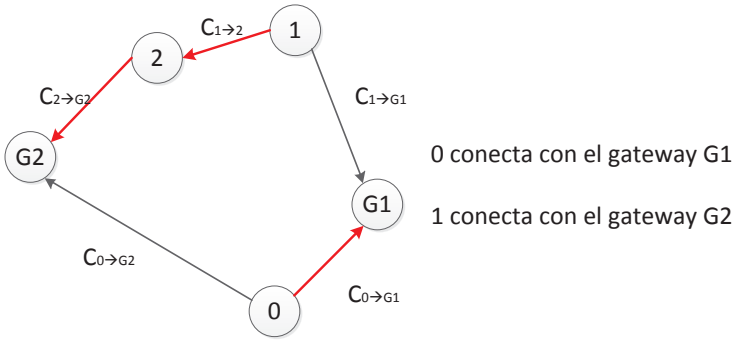


Figura 4.4: Conexión al mejor gateway

Como muestra la Figura 4.4 el nodo 0 tiene la opción de conectarse al G1 o al G2, pero debido a que $C_{0\to G1}$ es menor que $C_{0\to G2}$ se conecta al gateway 1. En el caso del nodo 1 el camino por 2 al gateway 2 es menor que el directo al G1, ya que $C_{1\to G1} > C_{1\to 2} + C_{2\to G2}$. Por lo que finalmente el nodo 1 se conectará con el gateway 2 a través de 2.

4.2.3. Dijkstra Modificado

Debido a ciertas características que se quería tener en cuenta en este trabajo, ha sido necesario realizar ciertas modificaciones al algoritmo original de Dijkstra. La primera de ellas, es añadir una limitación en el número de saltos que pueda tener una ruta. La segunda, surge a consecuencia de las pruebas realizadas con cambios de potencia durante el desarrollo del proyecto. A continuación se explican ambos cambios.

Restricción en el número de saltos

Como ya se ha comentado en la introducción de esta sección, para enriquecer los resultados de este trabajo, ha interesado añadir una restricción en el número de saltos máximo en una ruta origen-destino. Esto parece algo lógico, ya que no tiene mucho sentido que un paquete esté dando saltos por la red un número indefinido de veces lo que, además, no es eficiente.

Al añadir esta modificación se consigue que, cuando se están creando las rutas de mínimo coste, haya un contador del número de saltos que tendría esa ruta, por lo que si se supera este valor máximo, esta ruta se marcaría como no realizable y, si fuese posible, se buscarían otras alternativas, que, aunque tengan mayor coste, tienen menos saltos.

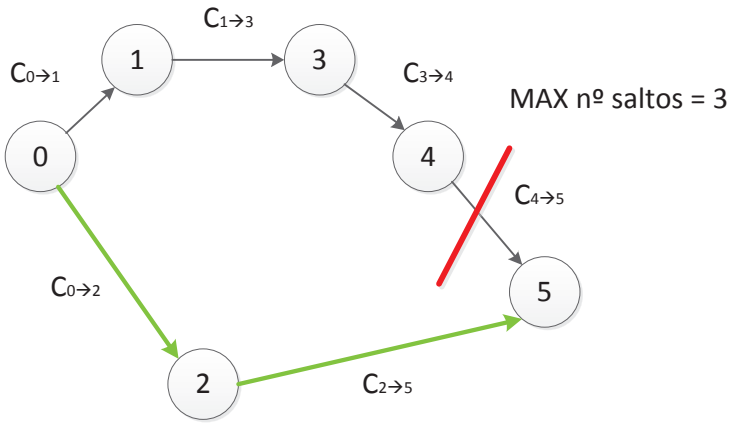


Figura 4.5: Restricción de saltos

Como demuestra la Figura 4.5, la ruta óptima en un principio podría, para ir del

nodo 0 al 5, $0-1-3-4-5$ ya que la suma de los pesos de esta ruta podría ser menor que la ruta $0-2-5$, pero al existir una restricción de saltos a 3, la primera ruta no es posible, ya que requiere de 4 saltos. Por lo que finalmente, se elige la ruta de más coste, pero que cumple la restricción.

Para implementar esta característica ha sido necesario el siguiente cambio en el algoritmo:

1. Se parte de la idea de que se tiene un grafo con N nodos, en el que se quiere conocer la distancia de X al resto de nodos. Para guardar estas distancias se inicializa un vector D con tamaño N y, para registrar los saltos, un vector H.

2. Todas las distancias del vector D son puestas a un valor infinito relativo, al igual que las de H, ya que al principio son desconocidas excepto la del propio nodo X, que se establece a 0 ya que la distancia y los saltos de X a sí mismo son 0.

3. Se denomina a al nodo actual en el análisis que, en la primera iteración, será el nodo del que queremos conocer los caminos mínimos, es decir X.

4. Se recorren todos los nodos adyacentes de a, llamaremos a estos v_i.

5. Si la distancia desde X hasta v_i guardada en D es mayor que la distancia desde X hasta a sumada a la distancia desde a hasta v_i:

6. *a*) Añadimos un salto al valor que había en H_a y lo ponemos en H_i

   b) Si el número de saltos en H_i es menor que el número máximo de saltos permitidos, sustituimos la distancia que había en D_i por la distancia desde X hasta a sumada a la distancia desde a, hasta v_i.

$$\begin{aligned} if(D_i > D_a + d(a, v_i)) \\ H_i = H_a + 1 \\ if(H_i \leq MAXHOPS) \\ D_i = D_a + d(a, v_i)) \end{aligned} \tag{4.1}$$

7. Se marca el nodo a.

8. Se toma como nodo actual a el nodo de menor valor de D, y se repite desde el paso 4, mientras existan nodos no marcados.

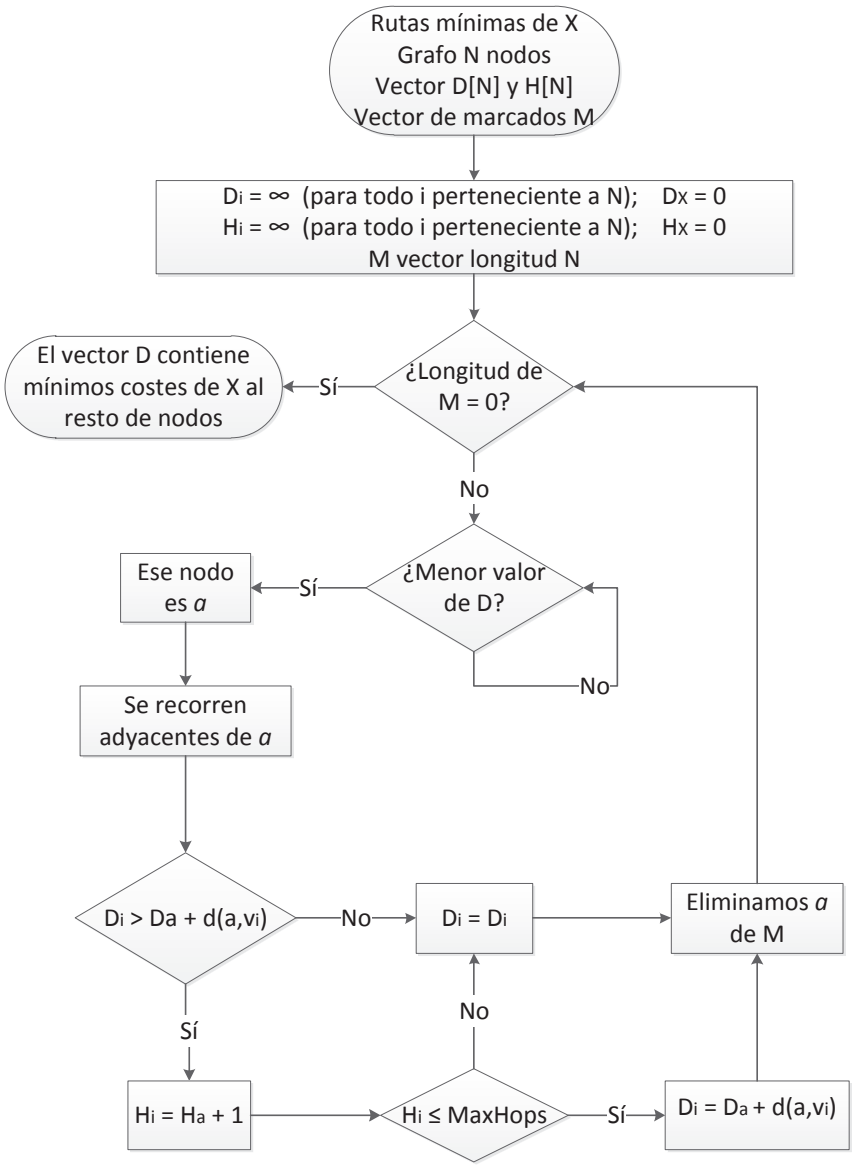


Figura 4.6: Diagrama de flujo de Dijkstra con restricción de saltos

Adaptación en potencia

Antes de describir las modificaciones en el algoritmo, se va a tratar de justificar los cambios llevados a cabo.

Para las simulaciones llevadas a cabo en este trabajo, se ha utilizado un tipo de tráfico UDP a nivel de transporte. Éste no usa ningún tipo de confirmación o ACK en el envío de sus tramas. Como ya se comentó previamente, el tráfico que se inyecta es unidireccional, del nodo al gateway. El problema es que, aunque a nivel de transporte UDP no necesite confirmación, a nivel 802.11 sí se envían ACKs. Esto daba lugar al problema que se ilustra en la Figura 4.7.

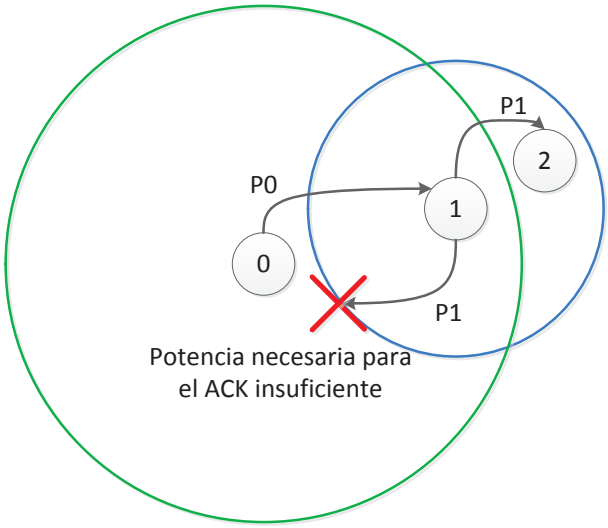


Figura 4.7: Fallo envío ACK

Según la tabla de rutas de cada nodo, se ve como el nodo 0, para llegar al nodo 2, debería usar como nodo intermedio el 1. Éste estaría en la corona A del nodo 0 y, por lo tanto, enviaría hacia ese nodo a una potencia P0. Lo mismo ocurriría con el nodo 1, que en su tabla tiene registrado que para llegar al nodo 2, el camino es directo y usa su corona B, pero con una potencia diferente P1, en este caso menor que P0. El problema aparece cuando el nodo 1 envía el ACK de capa de enlace hacia el nodo 0, ya que su potencia ha sido fijada para enviar al nodo 2 y, por lo tanto no es suficiente para alcanzar el nodo 0 con la confirmación, y el ACK se pierde. Así el nodo 0 retransmitiría el paquete una y otra vez pensando que no le ha llegado correctamente al nodo 1, cuando no es así.

Para solventar este problema se ha reconfigurado el algoritmo de enrutamiento. Lo que se ha hecho es que, cuando un nodo reciba un paquete, compruebe si la distancia del próximo salto es menor que la distancia de donde ha venido el paquete. Si es así, la potencia del nodo se tendrá que configurar a la mayor de las dos, en este caso el camino hacia atrás. La consecuencia es que la ruta podría dejar de ser óptima y se tendría que buscar otras

alternativas.

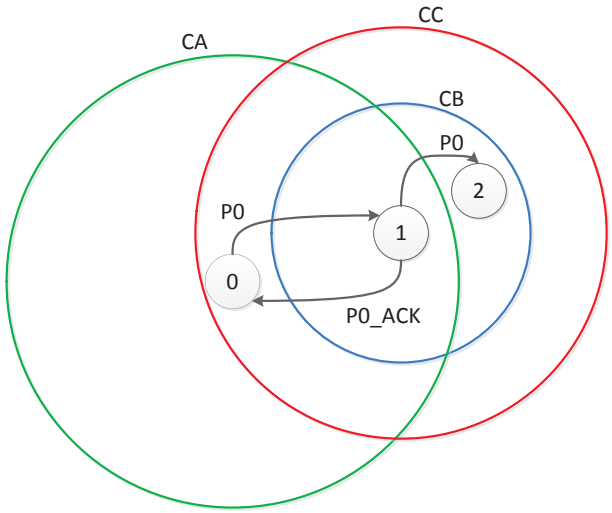


Figura 4.8: Corrección para recepción ACK

Como ilustra la Figura 4.8, aunque en el nodo 1 hubiera bastado con usar la corona B para llegar a 2, el ACK no habría sido recibido. Así que se utiliza la corona C, que tiene el mismo rango que la A, por lo que las potencias que se usan son las misma P0, para enviar el ACK y para llegar al nodo 2. Al usar una corona superior, el coste del enlace aumentaría y, como ya se dijo, podría hacer que el enlace no fuese óptimo.

Los cambios en el algoritmo se muestran a continuación.

1. Se parte de la idea de que se tiene un grafo con N nodos, en el que se quiere conocer la distancia de X al resto de los nodos, para guardar estas distancias se inicializa un vector D con tamaño N.

2. Todas las distancias del vector D son puestas a un valor infinito relativo, ya que al principio son desconocidas excepto la del propio nodo X, que se establece a 0.

3. Se denomina a al nodo actual en el análisis, en la primera iteración será el nodo del que queremos conocer los caminos mínimos, es decir X. Se crea una variable que tendrá el nodo en la ruta a a.

4. Se recorren todos los nodos adyacentes de a, llamaremos a estos v_i

5. Si la distancia desde X hasta v_i guardada en D es mayor que la distancia desde X hasta a sumada a la distancia desde a hasta v_i, y esta distancia es mayor que la distancia de a al previo de esa ruta, la distancia en D_i se sustituye por ésta última. En caso de la distancia de la suma ser menor que la del previo, el valor de D_i se

sustituye por la de la distancia al previo.

$$\begin{gathered} if((D_i > D_a + d(a, v_i)) \,\&\&\, (D_a + d(a, v_i) > d_{prev})) \rightarrow D_i = D_a + d(a, v_i) \\ else \rightarrow D_i = D_a + d_{prev} \end{gathered} \tag{4.2}$$

6. Se marca el nodo a.
7. Se toma como nodo actual a el nodo de menor valor de D, y volvemos al paso 4 mientras existan nodos no marcados.

En la Figura4.9 se muestra el diagrama de flujo del algoritmo anterior.

4.2.4. Variables útiles en escenarios Tcl

Este apartado recoge ciertos parámetros que pueden servir para modelar el tipo de escenario que se desea simular y realizar un estudio sobre él. Estas variables son fácilmente modificables a nivel usuario y están dentro del archivo .tcl principal, en el que se describe el escenario a simular. Estos parámetros se describen a continuación más detalladamente.

- **Rango de Cobertura:** Distancia máxima a la que podría transmitir un nodo. Si los demás parámetros de propagación como altura de antenas, umbrales de recepción y frecuencia, este parámetro modela la potencia necesaria para alcanzar la distancia máxima, a partir de esa distancia no es posible ningún paquete, ya que no es detectado por el nodo destino.
- **Número máximo de saltos:** Como su nombre indica, regula los saltos que puede dar un paquete en una ruta. Si se supera este número máximo se intentaría buscar una ruta alternativa de mayor coste pero con menos saltos; en caso de no encontrar esta opción el paquete se pierde.
- **Máximo coste de la ruta:** Este valor ajusta el coste máximo que puede tener una ruta entre dos nodos. Este valor no limita tanto como el número de saltos pero, al igual que el anterior, si se supera el máximo coste se intenta buscar otra opción aunque no sea la mejor.
- **Nodos con Tráfico:** Este proyecto ha utilizado, para las simulaciones, un único nodo transmitiendo, pero se ha dejado la opción de aumentar el número de nodos que puedan transmitir a la vez, para estudios posteriores de posibles colisiones e interferencias.
- **Número de Coronas:** Posiblemente, el parámetro más importante para este trabajo. Modela el número de coronas de las que ya se habló en la sección 4.2.1. Con

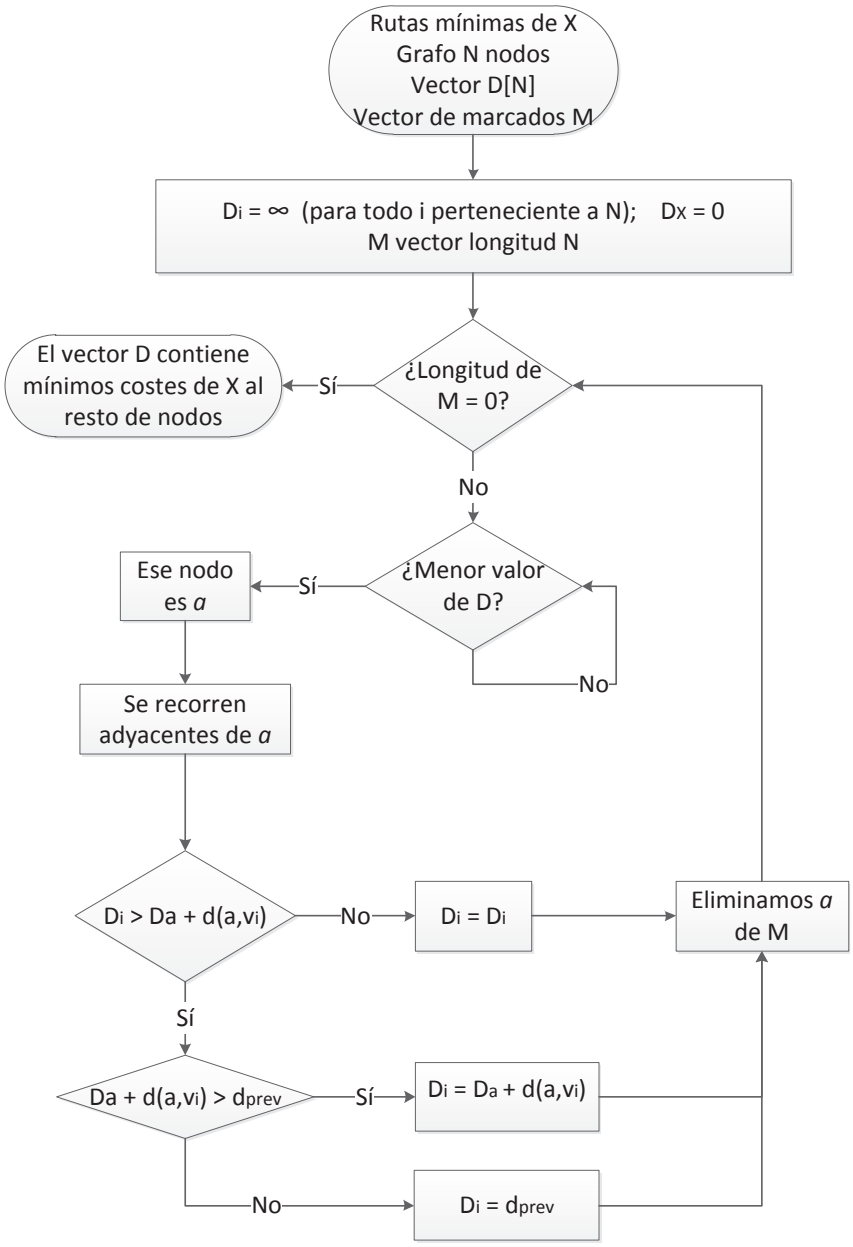


Figura 4.9: Diagrama de flujo de Dijkstra para reenvío correcto de ACK

ello también se modifica las posibles potencias a las que puede transmitir un nodo cuanto mayor sea el número de coronas más potencias a las que puede transmitir el nodo, siempre siendo la máxima la delimitada por el rango de cobertura. Además, los costes de los enlaces se definen teniendo en cuenta el número de coronas, por lo que mayor número de coronas proporciona mayor variedad de costes en los enlaces.

4.3 Modificaciones Wirelessphy

4.3.1. Modificaciones para resultados en potencia

Debido a que este proyecto intenta analizar la gestión de los recursos radio en redes malladas, más concretamente la eficiencia energética la capa más afectada es la física. De esta parte, en el Network Simulator, se encarga el fichero WirelessPhy. Por ello, para que el modelo descrito pueda implementarse, se han tenido que realizar unas modificaciones dentro del código de dicha clase.

En primer lugar, como se comentó en la sección 4.1, el modelo anterior al propuesto en este trabajo, utilizaba una potencia de transmisión determinada (y constante) desde un principio, y no se modificaba durante toda la simulación. Con el nuevo modelo es necesario poder tener una función capaz de modificarla cuando sea necesario. Para ello simplemente se ha cogido la variable que contiene la potencia y se ha modificado su valor con el que se pasaba desde las funciones del Static Routing.

La segunda modificación tiene que ver con la limitación tratada en la sección 4.2.3, en la que, debido a la modificación de la potencia, en ciertas circunstancias el ACK de nivel 802.11 no era recibido. Para paliar este problema, se han propuesto dos soluciones: la primera ya se ha tratado y consistía en la modificación del Dijkstra, con lo que dependiendo de si la potencia era mayor hacia atrás que hacia adelante en la ruta del paquete se tenía en cuenta la primera a la hora de transmitir. Esta opción es válida, pero no completamente dinámica, ya que se mantiene durante toda la transmisión una potencia que no tiene por que ser la óptima. Una mejor solución sería que, dependiendo del paquete que se recibiese o se fuese a mandar actuar consecuentemente. Por eso, cuando en este caso se reciba un ACK, sólo este se mandará a la potencia necesaria para llegar a su origen, en cambio si el paquete es el de datos y necesita continuar la ruta se enviará a la potencia correspondiente al siguiente salto. Con esto se hace consigue la mejor eficiencia energética posible para este modelo.

En la Figura 4.10 se muestra la implementación del aspecto tratado en el párrafo anterior; como se ve la potencia que se usa para el ACK y para el paquete de transmisión (forward) son diferentes, dependiendo de la corona en la que se encuentren los nodos correspondientes.

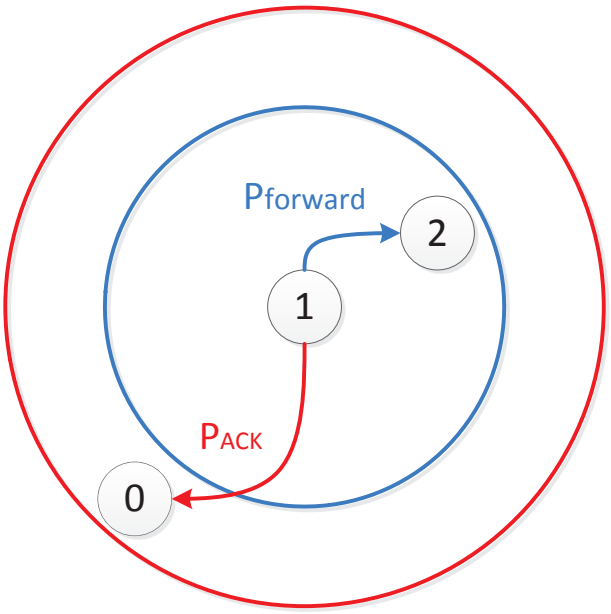


Figura 4.10: Gestión Dinámica de Paquetes

4.3.2. Modificaciones para resultados de calidad del enlace

La segunda parte del proyecto intenta comprobar si el mismo modelo de gestión dinámica de potencia se puede utilizar en escenarios donde existan pérdidas en el medio físico y si en estos casos se consigue una mejora o no.

Para ello y como se ha comentado en secciones anteriores, se ha utilizado el modelo de canal *SimpleModel*, cuyo funcionamiento se explicó en el Capítulo 2. Para poder funcionar con él es necesario introducir, en el momento de recepción, las líneas de código necesarias para que compruebe si el paquete es recibido correctamente o, por el contrario, si es erróneo. Dependiendo de los parámetros que se estableciesen en el tcl, la probabilidad de pérdida aumentará o disminuirá. En este apartado no se ha modificado la potencia que se envía a cada nodo, se ha tomado la máxima del rango y será el algoritmo de Dijkstra el que elija el camino mediante los pesos otorgados por el modelo de coronas. Esto se hace ya que el modelo de canal modifica su probabilidad de error dependiendo de la potencia que reciba el nodo receptor. Por lo tanto, si se sigue utilizando la gestión dinámica de potencia es probable que, al ajustar la potencia de transmisión para ahorrar energía, se esté perdiendo calidad en el enlace. Este punto de estudio no se llevará a cabo en este trabajo sino que quedará como posible línea futura para buscar un compromiso entre ahorro energético y calidad del enlace.

5

Simulaciones y Resultados

Una vez adaptadas todas las modificaciones necesarias para el funcionamiento de la gestión dinámica: canal *Simple Model*, *Static Routing*, el Monitor y los cambios en la capa física, en este capítulo se describe la operación conjunta de todas estas funcionalidades para conseguir los resultados que se esperan de este estudio.

Se va a ejecutar un número determinado de simulaciones en diferentes escenarios, desplegados de manera aleatoria, desde el punto de vista de posición de los nodos, además se escogerán, mediante diferentes metodologías, los nodos que utilizarán el role de *gateway*. Debido a la gran variedad de escenarios posibles, es necesario disponer de un mecanismo de automatización de las simulaciones.

Como consecuencia del gran número de casos posibles, se fijará el valor de ciertos parámetros, para así conseguir un número de resultados coherentes al estudio. Las consideraciones que se han tomado son las siguientes:

- Se utilizarán tres casos de coronas, para 1, que refleja la operación tradicional, y para 3 y 5 coronas.
- El tamaño del escenario será de 100m x 100m en el que se desplegarán 80 nodos de los cuales se irá variando el número de gateways de 5 a 10 posibles.
- Modelo de Canal: Para los casos de análisis de potencia se usará el modelo *TwoRay-Ground*, en el que los paquetes llegan a su destino, sin pérdidas durante la transmisión (siempre que estén en su área de cobertura), en cambio, para el caso de medidas de calidad del enlace se usará el *Simple Model*, del que ya se habló en la Sección 2.4.1, con valores $\alpha = 0{,}33$, dmax = 15 y $\beta = 1$.
- A nivel de transporte, se ha inyectado un tráfico de 1472 *bytes/paquete* con un intervalo entre paquetes de 2 msec. Con esto se satura el canal, ya que el rendimiento máximo de 802.11b (con un único salto) se situa 6Mbps. Cada simulación dura 60 segundos.

- Para el caso de medidas de calidad, no es tan interesante saturar el canal si no la comparativa entre los diferentes casos, por lo que se ha escogido un tráfico con el tamaño de paquete 1000*bytes* y ahora lo que variamos es el *rate* entre 1Mbps y 5Mbps en pasos de 200Kbps con un máximo de 7500 paquetes. Se van a representar los resultados para estrategias de 1 y 3 coronas con límite de saltos de 3,4 y 5.
- Como ya se dijo, sólo un nodo envía tráfico en cada simulación, ya que si se generará más tráfico de fondo, se aumentaría considerablemente el tiempo de simulación y no aportaría información a los resultados que se buscan.
- Se establece finalmente un máximo de 3 saltos para la exploración de las rutas del algoritmo.

5.1 Herramienta de Automatización de las Simulaciones

Consiste en un conjunto de ficheros ejecutables, C++, Perl y archivos de configuración que permiten realizar fácilmente muchas simulaciones variadas pudiendo modificar: número de nodos y sus configuraciones, número de simulaciones, tamaño del escenario, variables de la gestión dinámica, canal, etc.

Primero se va a realizar un resumen de los principales puntos que tiene una única simulación y posteriormente se explicará como se lleva a cabo las modificaciones pertienentes para poder llevar a cabo muchas simulaciones. Además hay que tener en cuenta que se están simulando 4 tipos de escenarios diferentes explicados en la sección 2.2, por lo que también se verá en qué afectan éstos al método de simulación.

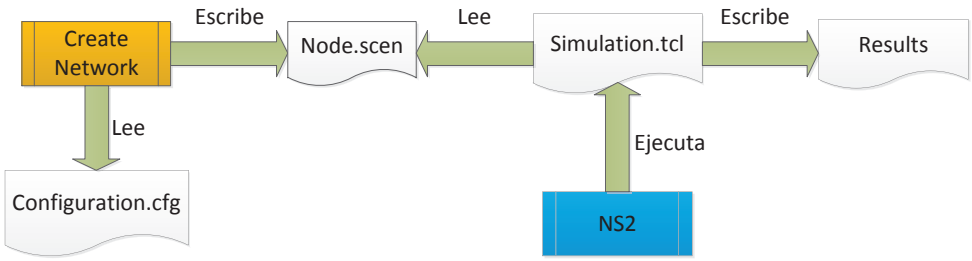


Figura 5.1: Ejemplo de una simulación

La Figura 5.1 muestra los elementos de los que consta una única simulación, la función de cada uno de estos se describe seguidamente.

- **Configuration File:** Archivo de texto modificable por el usuario y que contiene información del escenario como tamaño, número de nodos, número de gateways, rango de cobertura, etc.

- **Ejecutable Create Network:** Lee el archivo de configuración y crea automáticamente el escenario con los nodos desplegados, guardándolo en el archivo *nodes.scen*. Los nodos son colocados de manera aleatoria sobre todo el escenario.
- **Archivo tcl de simulación:** Aglutina todos los parámetros necesarios para la simulación, protocolos, tipo de canal, instancia el monitor, acceso a las variables y archivos de salida posibles. Mediante la lectura del *node.scen* inicializa (tcl) los nodos y sus posiciones.
- **Fichero de resultados:** Guarda los resultados de las simulaciones
- **Simulador NS:** Ejecuta el archivo tcl para iniciar la simulación.

Con esto ya se podría realizar una simulación única. Pero en el caso de este trabajo, para conseguir resultados se han realizado muchas ejecuciones por escenario por lo que este método no es del todo eficiente. Aquí es donde entra en escena el script Perl, con lo que se puede conseguir modificar los archivos tcl, mediante algunas variables más en el archivo de configuración, sin necesidad de modificar los archivos C++. Por último permite hacer llamadas a ejecutables con lo que se cubren los 4 pasos anteriormente descritos con un solo archivo.

Su principal ventaja es que permite realizar simulaciones repetidamente. Puede hacer barridos en cualquier parámetro que sea interesante: gateways, coronas... Además, se han añadido un conjunto de variables de control, para escoger si se desea guardar o no los resultados y los escenarios, usar despliegues concretos y mostrar en consola la evolución de la simulación. La herramienta quedaría finalmente como muestra la Figura 5.2.

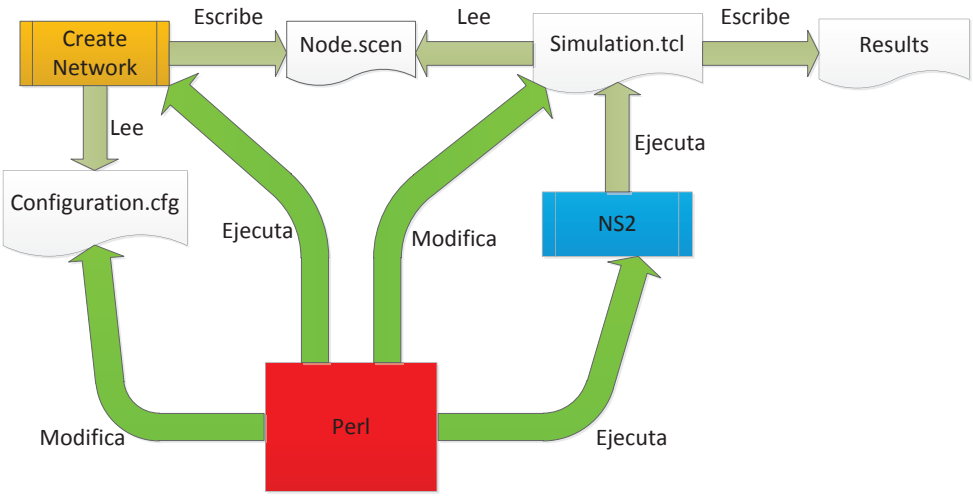


Figura 5.2: Herramienta completa

La secuencia de cada iteración recorre el número de nodos, el número de gateways, el número de coronas y, por último, el número de simulaciones, cambiando los parámetros

necesarios en los archivos de configuración, llamando a Create Networks para generar los escenarios y ejecutando el archivo tcl para empezar la simulación. Por último guarda los resultados temporalmente y los borra o no dependiendo de lo indicando en el script Perl.

Finalmente, teniendo en cuenta que las estrategias de despliegue 3 y 4 utilizan para conseguir los despliegues óptimo y sub-óptimo, un ejecutable llamado *Popstar*, que determina qué nodos deberían ser Gateways es necesario modificar el procedimiento anterior, como se ve en la Figura 5.3

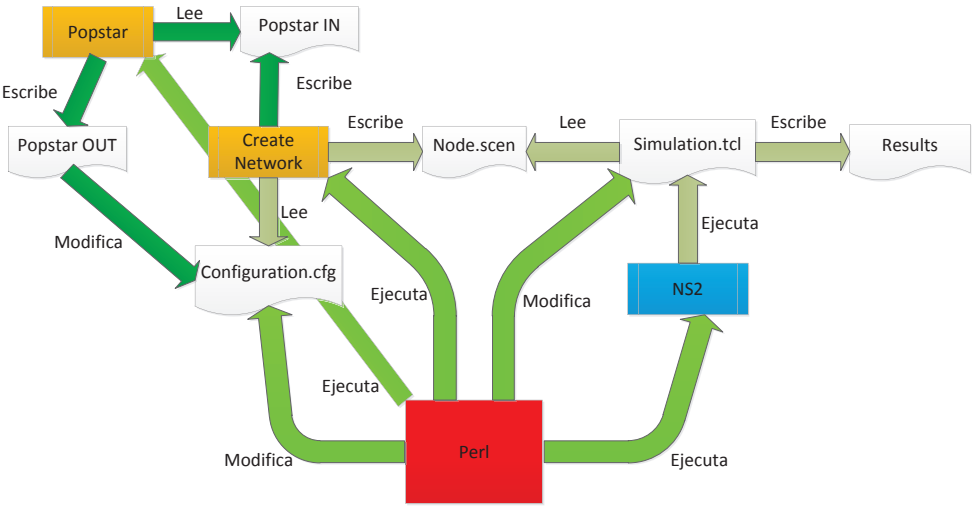


Figura 5.3: Herramienta completa con Popstar

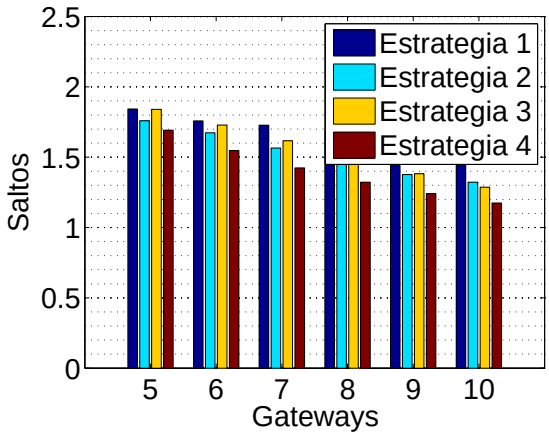
5.2 Análisis del número medio de Saltos

Este apartado presenta los resultados obtenidos en las medidas de las diferentes estrategias, en cuanto al número de saltos medio, utilizando 100 ejecuciones independientes por simulación. Además, se ha variado el número de gateways que hay en el escenario y el número de coronas en que se ha dividido el rango de transmisión de los nodos. Por último, se ha dividido el análisis teniendo en cuenta el Dijkstra normal o el modificado para los casos de posible pérdida de ACK.

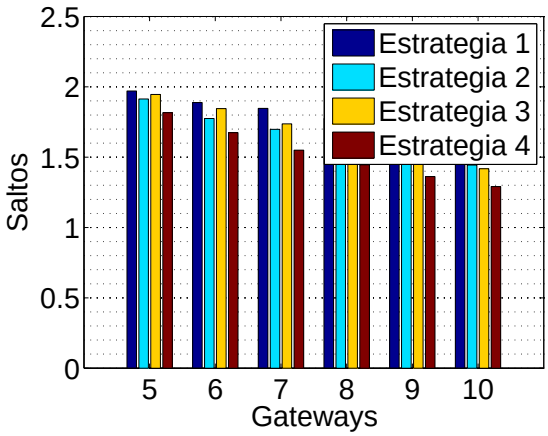
5.2.1. Análisis normal

De la Figura 5.4 se pueden extraer las siguientes conclusiones:

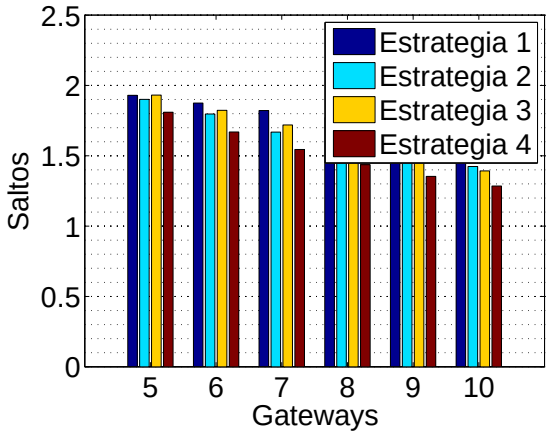
- El primer elemento a destacar, es que como se aprecia en las imágenes en ningún caso se superan los 2 saltos de media en ninguna estrategia, para ningún tipo de corona



(a) 1 Corona



(b) 3 Coronas



(c) 5 Coronas

Figura 5.4: Número de saltos medio dependiendo del número de coronas

e independientemente del número de gateways. Esto se debe a que, debido a la gran población de nodos que tiene el escenario, es bastante sencillo encontrar un punto de conexión cercano, siempre entre 1 y 2 saltos. Además, limitando el número de saltos a 3, posibles rutas más largas con 4 o 5 saltos; serán descartadas. Es posible pensar que en el escenario 1, completamente aleatorio puedan darse estas situaciones.

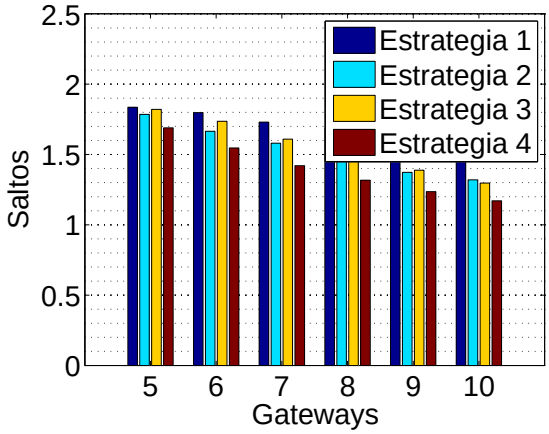
- Se ve fácilmente que para todos los casos el número de saltos disminuye cuando va aumentando el número de gateways. Esto tiene lógica. A medida que se incrementan los puntos a los que los nodos pueden conectarse, es más sencillo encontrar uno más cercano y, como consecuencia, el número de saltos necesarios para alcanzar un gateway es menor.

- El segundo aspecto a destacar es que para cada estrategia va disminuyendo el número de saltos. Esto es debido a que en la primera estrategia todo el despliegue es aleatorio, tanto gateways como nodos, por lo que puede darse el caso de que varios gateways queden muy cerca entre sí y estén más alejados de ciertos nodos, con el consecuente aumento del número de saltos para conectarse a ellos. En el segundo escenario el despliegue es el óptimo geométrico, por lo que se cubre casi todo el área posible, separando de manera equidistante los gateways. Entre el primer caso y éste se aprecia una caída importante en el número medio de saltos. Para las estrategias 3 y 4, que utilizan una búsqueda óptima de la localización de los gateways, son los mejores casos. Los nodos encuentran siempre cerca un gateway al que conectarse casi a distancia de un salto, lo que va disminuyendo más cuantos más gateways añadamos. En el caso de la estrategia 3, con menos gateways aumenta el número medio de saltos porque ésta coloca más gateways en los subgrafos mayores, por lo tanto, zonas más lejanas tendrían que utilizar más saltos para la conexión. En cambio, la estrategia 4 intenta cubrir el mayor número de subrafos posible con los gateways de los que dispone, así que, nodos más aislados podrían tener gateways más cercanos y en subgrafos más poblados habría menor número de gateways.

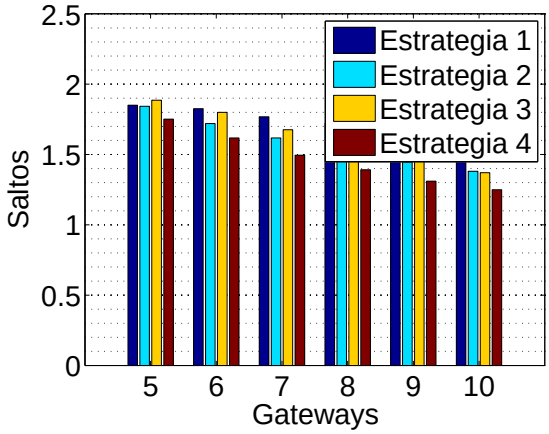
- Finalmente, comparando cada una de las gráficas, la que menor número medio de saltos ofrece es el caso de 1 corona, algo lógico ya que el coste que proporciona cada nodo es único y, aunque el gateway estuviese a mayor distancia que otra posible ruta de más saltos, no la escogería, ya que el coste sería el mismo y el algoritmo de Dijkstra la desecharía. Para los casos de 3 y 5 coronas los resultados son muy parecidos, en principio cabría pensar que para el caso de 5 coronas el número de saltos medio aumentaría, ya que la variedad de costes es mayor, el radio de las coronas menor y por lo tanto podrían aparecer rutas más largas con menos coste. Pero, debido a la gran densidad que se tiene en el escenario creado, la cercanía entre nodos hace que casi siempre haya un gateway a una distancia de 1 o 2 saltos, con lo que la media no aumenta.

5.2.2. Análisis Dijkstra modificado

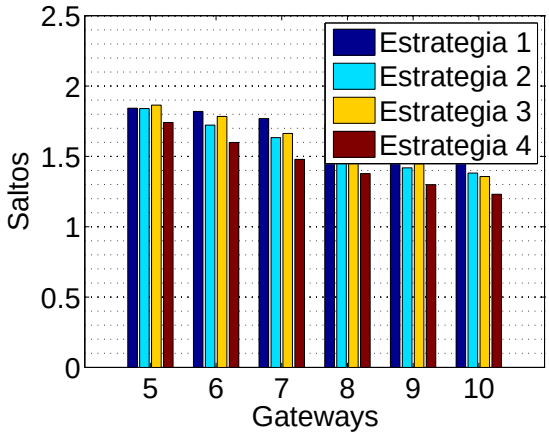
Ahora se realiza el mismo análisis que el anterior pero para el caso de la modificación del algoritmo de Dijkstra, comentado en la Sección 4.2.3.



(a) 1 Corona



(b) 3 Coronas



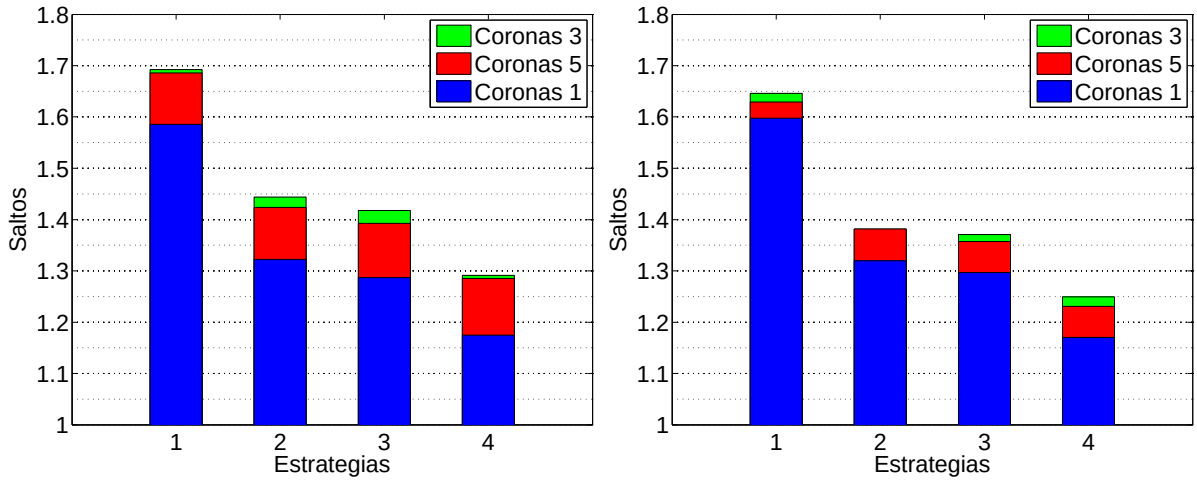
(c) 5 Coronas

Figura 5.5: Número de saltos medio dependiendo del número de coronas con el algoritmo de Dijkstra modificado

De las gráficas mostradas en la Figura 5.5, en comparación con las anteriores, a penas se aprecian cambios. Como ya se mencionó anteriormente, debido a la proximidad entre nodos por la alta densidad del escenario, el caso de pérdida del ACK no es tan probable como en el de nodos aislados y con más distancia entre ellos, lo que explica la escasa diferencia de los resultados de ambas configuraciones.

5.2.3. Comparativa

A continuación se muestran dos gráficas en las que se hace un análisis desde otro punto de vista, en el que sólo se escoge el mejor escenario visto en los anteriores casos, que sería para 10 gateways, y se comparan las diferencias entre las estrategias y el uso de 1,3 ó 5 coronas. Una de las gráficas muestra el desarrollo normal y el otro con el algoritmo modificado.



(a) Análisis normal

(b) Dijkstra modificado

Figura 5.6: Número de saltos medio para el caso de un escenario con 10 gateways

Básicamente como ya se comentó en las secciones anteriores, se observa una disminución en el número de saltos para todas las coronas en función de la estrategia de asignación de gateways. Lo más destacable es en la comparación de los casos de 1 corona, 3 y 5. Aquí, para el análisis normal, las diferencias son bastante significativas, al aumentar el número de coronas. En cambio, en el caso del Dijkstra modificado, esta diferencia es menos relevante, lo que podría deberse al cambio que se realizó para evitar la pérdida del ACK, lo que modifica ciertos pesos de alguna ruta con la consecuencia de buscar otra ruta óptima alternativa con más peso y menor número de saltos, por lo que las diferencias disminuyen, aunque sigue sin ser muy significativa, lo que quiere decir que los casos en lo que ocurre este efecto no son demasiados.

Por otro lado, el caso de 5 coronas da un número medio de saltos ligeramente menor que para el caso de 3 coronas. Esto puede deberse a que la limitación de saltos a 3 elimina posibles caminos de 4 o 5 saltos, más probables con 5 coronas, con la consecuente necesidad de buscar rutas alternativas con más coste y menos saltos.

5.3 Análisis de Potencia

Después de los resultados del estudio del número de saltos, se va a realizar una comparación que de interés en el marco de este trabajo, en cuanto a las mejoras que puede introducir el modelo creado en términos de eficiencia energética.

Para el análisis realizado se ha utilizado el mismo caso que en la comparativa de saltos medio, el escenario 4 con 10 gateways. En principio se usa este escenario porque es el que mejores prestaciones ofrece.

Se han comparado, como en el apartado anterior, los resultados del Dijkstra modificado y del algoritmo sin modificar. Además, para ver bien las diferencias, se ha normalizado la potencia, escogiendo como valor de normalización el punto de máxima potencia y comparando los demás sobre él. También se muestran los resultados para los casos de 1, 3 y 5 coronas.

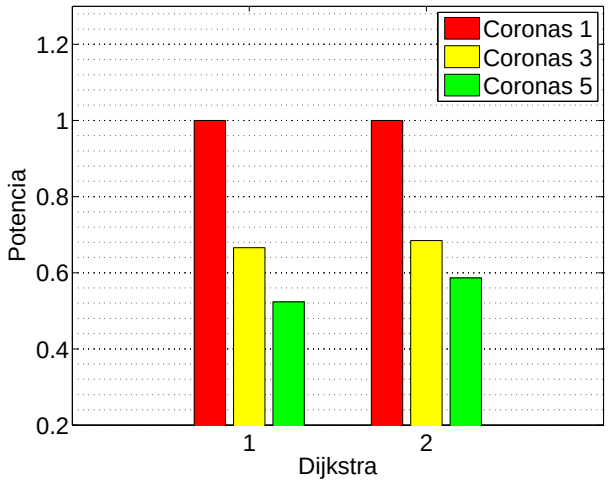


Figura 5.7: Comparación de potencia

A la luz de los resultados de la Figura 5.7, se puede llegar a la conclusión de que para el caso de 1 corona, como era de esperar, se consiguen los peores resultados, es decir se obtiene el mayor consumo de potencia. Esto se debe a que sería el caso inicial del que se habló en el Capítulo 4, donde se envía la máxima potencia en todos las situaciones, sin haber una gestión dinámica, por lo que independientemente de que los gateways a los que se conectan los nodos pueden estar cerca, la potencia que se usaría sería la máxima posible para alcanzar todo el rango de cobertura. Aquí tampoco hay diferencia entre usar el Dijkstra normal y el modificado; de hecho, para el caso de una corona, el problema del fallo de envío de ACK no existe, ya que si el nodo A está en el rango de B, B estaría en el de A y, por lo tanto, siempre llegaría el ACK.

A partir de aquí se comprueba que el modelo claramente mejora la operación tradi-

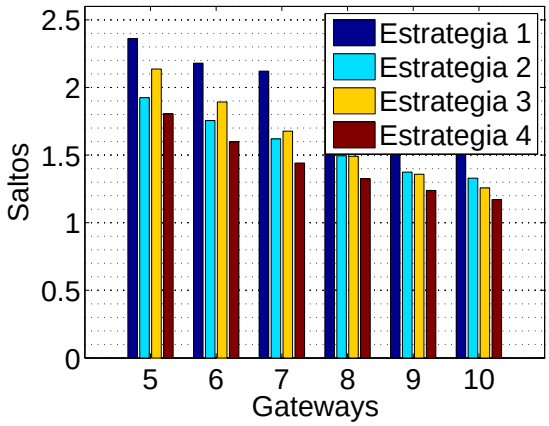
cional por defecto. Para el caso de 3 coronas la mejora es mayor del 30 % y, para el caso de 5 coronas se estaría hablando de una mejora del 50 %. Todo esto para el algoritmo sin modificar y con la gestión dinámica a nivel de paquete. La mejora sustancial se debe a que hay más posibilidades para enviar a diferentes potencias y, cuanto mayor sea la cercanía al nodo menor será esa potencia. Además, como se vio en el análisis previo el número de saltos medio no superaba los dos en ninguno de los escenarios, por lo que en las rutas de uno o dos saltos, los mejores resultados se obtienen con 5 coronas en la mayoría de los casos, porque la potencia que se envía sería menor que con las otras configuraciones. Finalmente, se pone de manifiesto un comportamiento parecido para el caso del Dijkstra modificado, aunque las mejoras son menores, porque cuando es necesario usar el método para que no se pierda la confirmación, la potencia que se usa es mayor a la que se usaría en el mismo escenario pero con la gestión dinámica de paquete.

5.4 Análisis de la calidad del enlace

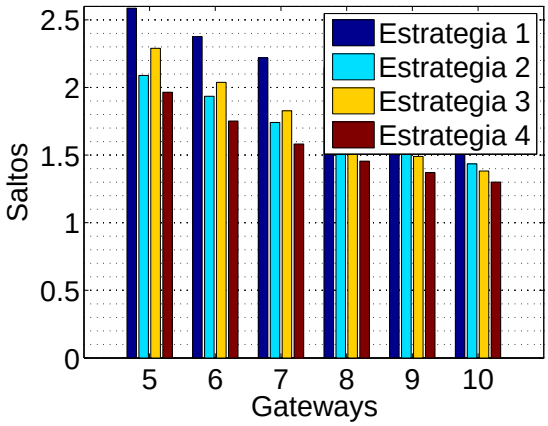
Por último, se van a recoger un conjunto de medidas sobre la calidad del enlace, usando ahora el modelo de canal con pérdidas Simple Model. Se caracterizará el throughput para el caso del escenario sub-óptimo topológico y con 10 gateways, que parece ser el que mejor resultados obtiene para eficiencia energética.

Primero se volvió a realizar el análisis del número de saltos, modificando el máximo número de saltos por ruta a 5, para que ciertas alternativas que debido a la limitación anterior pudieron ser rechazadas, ahora se tengan en cuenta.

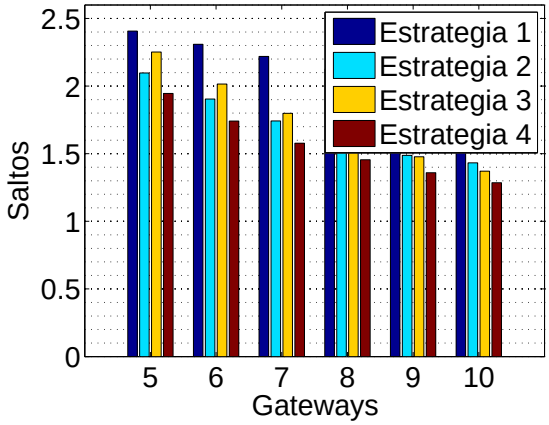
En la Figura 5.8 se muestran los resultados de la modificación del máximo número de saltos permitido a 5, en vez de a 3 como en los resultados anteriores. La estrategia de despliegue aleatorio aumenta el número de saltos medio considerablemente. Esto es normal, ya que los gateways pueden aparecer en cualquier punto del escenario y es necesario emplear más saltos para que los nodos se conecten al gateway óptimo. Además, para todos los casos, tanto de 1, 3 ó 5 coronas también ha habido un aumento generalizado de la media, sobrepasando ya en algunos casos los dos saltos para realizar la conexión. Por lo tanto la suposición de que ciertas rutas estaban limitadas por el número de saltos es correcta. También se observa que el escenario 2, el caso óptimo geométrico, se comporta bastante bien en cuanto a número de saltos se refiere, algo esperado, ya que se busca dar la mayor cobertura posible con los gateways de que se dispone. La estrategia 3, mejora como en casos anteriores, cuanto mayor número de gateways haya. La elección de la posición del gateway abarca más subgrafos a medida que los incrementamos, disminuyendo el número de saltos necesario para que un nodo se conecte a él. Por ello, con números bajos de gateways se percibe en este escenario, un aumento de su media, quedando por detrás de estrategias como la 2 ó la 4. En cambio cuando el número de GWs es mayor, la estrategia se comporta mucho mejor. Por último, la mejor estrategia para este análisis sigue siendo la



(a) 1 Corona



(b) 3 Coronas



(c) 5 Coronas

Figura 5.8: Número de saltos medio dependiendo del número de coronas para número máximo de saltos 5

4, el despliegue subóptimo, que sigue alcanzado un número medio de saltos muy parecido al de los casos ya comentados.

Como se hizo anteriormente, para todas las estrategias se realiza una comparativa fijando el número de coronas a 5 y el máximo número de gateways a 10. La Figura 5.9 recoge los resultados de esta simulación. Se comprueba que claramente la estrategia 1 presenta un comportamiento muy malo en relación al número de saltos, incluso en el caso de mayor población de gateways, alcanzando casi los dos saltos por conexión. Las demás estrategias no difieren mucho de su comportamiento para el número máximo de 3 saltos. Para todas las estrategias, la configuración de 1 corona sigue siendo la que menor número medio de saltos ofrece.

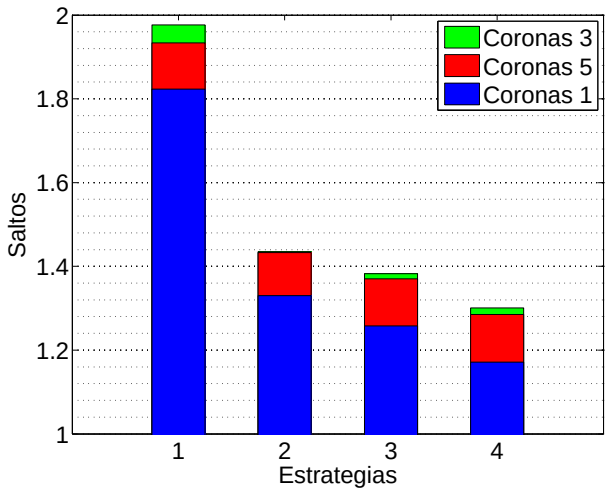
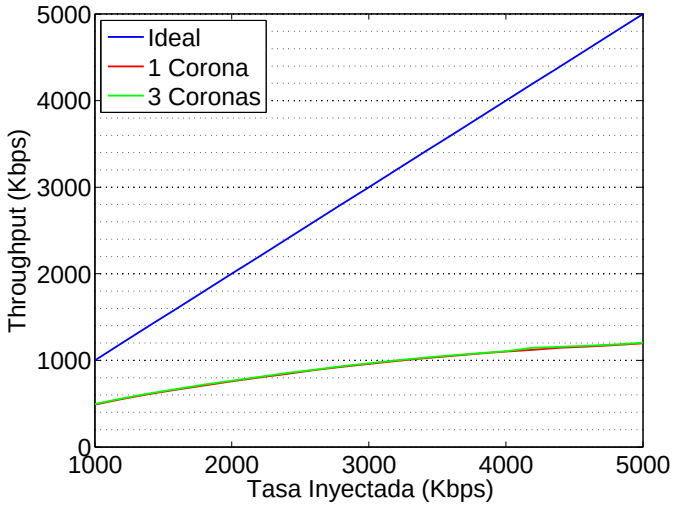
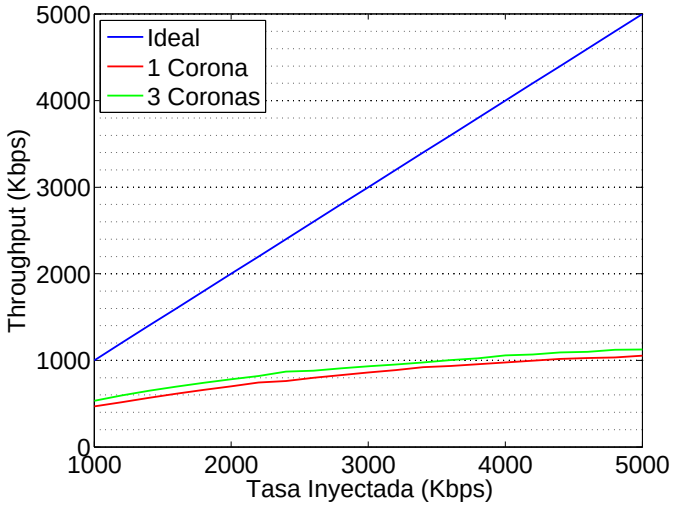


Figura 5.9: Comparación nº medio de saltos para escenario 10 gateways y 5 saltos máximo

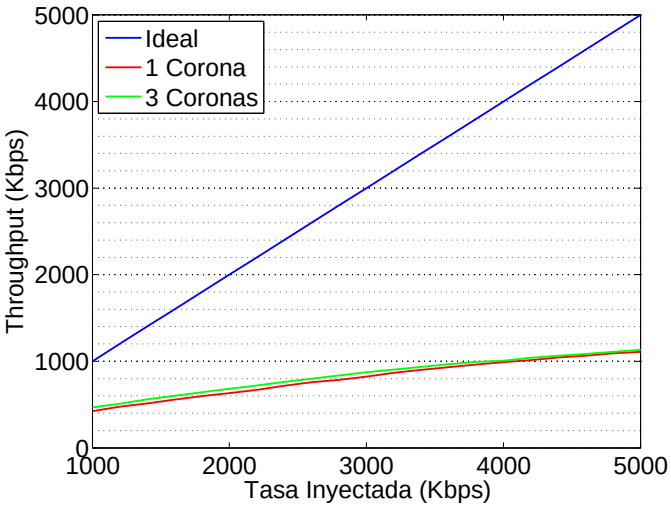
A la vista de los resultados anteriores, se pasó al análisis del escenario concreto que se comentó al principio de la sección. Se va a realizar una observación de los casos de 1 y 3 coronas con 100 simulaciones para cada caso, y con un rango de tasa binaria entre 1Mbps y 5Mbps con un máximo de 7500 paquetes transmitidos, con lo que se extraerán unas gráficas donde se observará la saturación del canal y el comportamiento del modelo de coronas. Además, se modificará el número máximo de saltos de 3 a 5, para poder observar si se llega a un límite de rendimiento para el modelo o qué caso puede dar mejores resultados. El no incluir mayor granularidad en las coronas es debido a que, cuando se usa el modelo de canal *Simple Model* los enlaces situados de manera física a una distancia menor de $\alpha * d_{rangocobertura}$ no tienen pérdidas de paquete. Como $\alpha = 0{,}33$ y el rango es de 15 metros, este rango sin pérdidas se produce para distancias menores de 5 metros, coincidente con la primera corona del caso de 3 coronas. Por lo que, si se consigue que las rutas usen mayor número de primeras coronas, habrá un número pequeño de pérdidas. El aumentar el número de coronas, sólo dividiría las zonas de las que se está compuesto el modelo de canal en más trozos con diferentes costes, pero no añadiría información útil a los resultados.



(a) 3 Saltos



(b) 4 Saltos



(c) 5 Saltos

Figura 5.10: Medida de Throughput usando modelo de coronas

A la vista de las gráficas de la Figura 5.10, se puede concluir que el caso donde mayor throughput se obtiene es con la limitación máxima de 4 saltos, además de ser el caso en el que el modelo de coronas presenta más diferencia con el modelo tradicional. Se observa que, a medida que se aumenta la tasa binaria, el throughput se incrementa pero no de manera lineal, ya que cada vez se separa más del caso ideal, lo que indica que, posiblemente se estén produciendo pérdidas de paquetes. En cuanto al modelo usado de coronas, se comprueba que las diferencias entre los casos son pequeñas, ya que en el escenario escogido (Estrategia 4, 10 gateways) el número medio de saltos se situaba alrededor de 1.5, por lo que la gran mayoría de las conexiones con el GW se realizan de manera directa o, como mucho apoyándose de un nodo más, no importando demasiado si se usa 1, 3 ó 5 coronas. En la Figura 5.11 se muestra un escenario desplegado de los que se han usado para realizar las simulaciones pertinentes. Como ya se comentó se ve que la mayor parte de las conexiones a los gateways (puntos azules) son directas o con un salto intermedio. En rojo sería el caso de usar un modelo de 1 corona y en verde el caso de 3 coronas. Se ve que, aparentemente, las dos estrategias suelen elegir el mismo camino en todas las conexiones. Con esto se pone de manifiesto la explicación a las escasas diferencias entre las configuraciones de 1,3 ó 5 coronas.

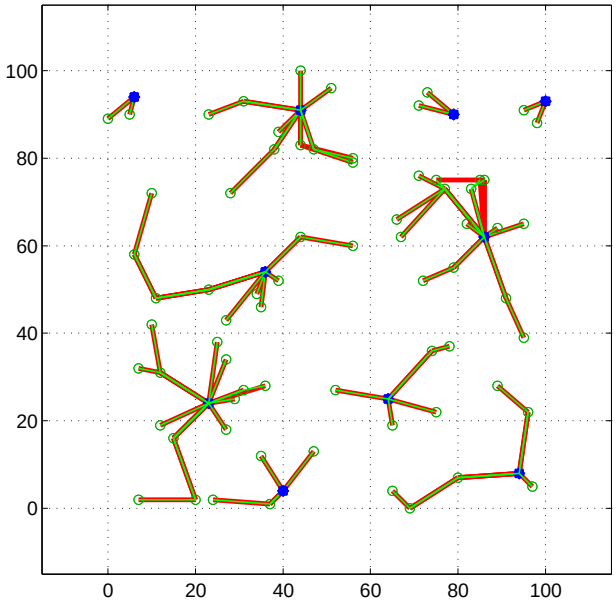
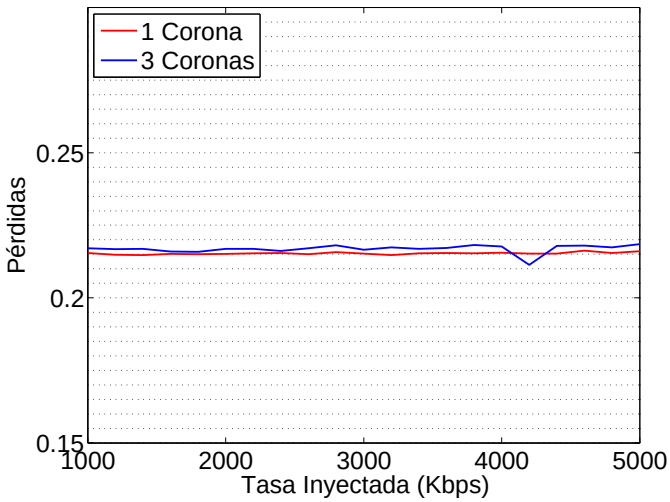


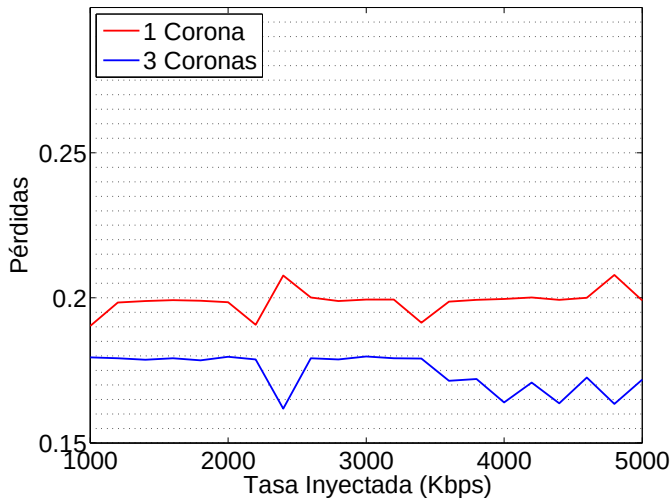
Figura 5.11: Ejemplo de despliegue de un escenario

Otra forma de poder representar el rendimiento de la ruta es mediante el tanto por uno de paquetes perdidos, es decir, el cociente entre el número de paquetes que no llegan a su destino entre el número de paquetes totals enviados, en este caso se sabe que son 7500. Con esta medida se intenta sacar más conclusiones de la aportación del nuevo modelo a la característica calidad del enlace. La figura 5.12 muestra este coeficiente para tasas entre 1Mbps y 5Mbps para los casos de 3,4 y 5 saltos y el uso de 1 ó 3 coronas. Gracias a este análisis, se pueden sacar nuevas conclusiones. En primer lugar, se confirma que el

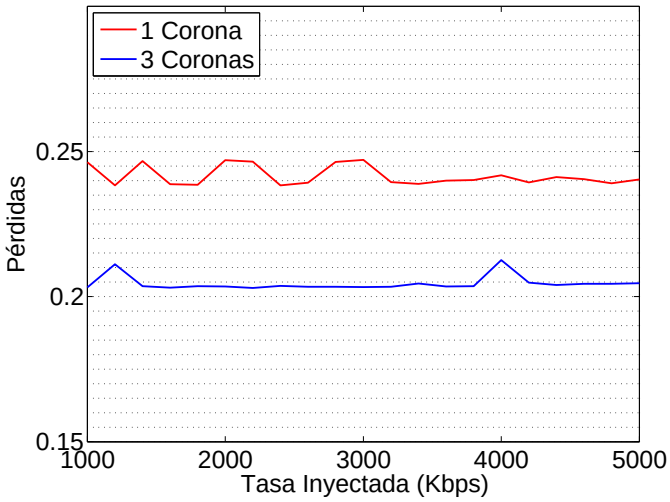
mejor caso sigue siendo el de un máximo de cuatro saltos teniendo pérdidas inferiores al 20 % tanto para 1 como para 3 coronas, aunque el caso de 3 coronas consigue pérdidas inferiores 18 % mejores que las obtenidas para 1. El caso de 3 saltos en cambio presenta los peores resultados en cuanto al modelo que se usa, ya que aunque el cociente es muy parecido entre 1 y 3 coronas, siempre suele estar por encima el caso de 3 coronas, por lo tanto para éste, el modelo no mejora las prestaciones, pero tampoco se puede decir que las empeore. Por último el caso de 5 saltos tiene la mayor diferencia entre 1 corona y 3. Para el tradicional las pérdidas rondarían el 25 %, mientras que gracias al nuevo modelo estas pérdidas disminuyen al 20 %.



(a) 3 Saltos



(b) 4 Saltos



(c) 5 Saltos

Figura 5.12: Pérdida de paquetes usando modelo de coronas

6

Conclusiones y Líneas Futuras

Este capítulo realiza un pequeño resumen de las ideas que se han extraído a partir de los resultados obtenidos al finalizar este trabajo. Posteriormente, y durante la realización del proyecto, se han llegado a plantear nuevas líneas de trabajo que podrían optimizar o mejorar los resultados ya obtenidos; éstas se resumen en el segundo epígrafe de este capítulo.

6.1 Conclusiones

Este trabajo se ha centrado en el análisis de gestión de recursos en redes malladas multi-salto sobre tecnología 801.11, buscando una eficiencia en términos energéticos e intentando modular al máximo la potencia de transmisión. Este beneficio energético podía tener como consecuencia un empeoramiento de la calidad del enlace, por lo que también se realizó un estudio de esta característica.

Para poder llegar a la consecución de estos objetivos se llevo a cabo una serie de pasos que se detallan a continuación:

- Se ha realizado un estudio y la lectura de varios artículos sobre gestión dinámica de potencia y eficiencia energética en redes malladas.
- Para poder usar el simulador NS2, se realizaron una serie de tutoriales y modificaciones en su estructura interna para adecuarlo al uso de múltiples interfaces y poder introducir el protocolo de enrutamiento estático.
- Dentro del *Static Routing*, se prepararon ciertas funciones capaces de modificar la potencia dentro de la capa física mediante el uso de parámetros como la distancia o la potencia en sí.

- Se necesitaba una entidad omnisciente para poder conocer todos los nodos de la red y crear las rutas que usará el Static Routing. Se diseñó e implementó el *Monitor*, con lo que, al crear el escenario, se registraban todos los nodos. En esta entidad se realizaban las operaciones deseadas para, después comunicárselas al agente de enrutamiento.

- Para la generación de rutas se utilizó el algoritmo de Dijkstra, que utiliza unos pesos asignados a los enlaces para encontrar las rutas de mínimo coste desde un nodo a los demás.

- Para asignar los costes se creó el modelo de Coronas que, además resulta útil para modificar el modelo actual de potencia única para todo el rango de transmisión y poder gestionar de manera dinámica esta variable. Dependiendo de la corona en la que se encuentren los nodos adyacentes se asignará un coste u otro al enlace correspondiente.

- Interesaba que los nodos se conectaran a un gateway y, en el caso de poder acceder a varios, que la conexión fuese con el de mínimo coste, por lo que se implementó la función necesaria para poder llevar a cabo este método.

- Se modificó el algoritmo de Dijkstra para poder limitar el número de saltos de las rutas, y que fuese capaz de buscar otra posibilidades. Otra modificación fue la reasignación de los costes si surgía el problema del envío de ACK.

- Tras configurar adecuadamente el Monitor, se realizaron los cambios pertinentes en la capa Física (*WirelessPhy*), para que se adaptase a las acciones que se querían realizar desde el Monitor y que pasaban a través del Static Routing. Por ejemplo, se estudió el caso de la gestión dinámica de paquetes en la que, dependiendo del tipo de paquete, se transmitiera a una potencia u otra.

- Para las simulaciones se usaron 4 tipos de estrategias diferentes de despliegue de los gateways. Con estos escenarios se llevaron a cabo las simulaciones de número medio de saltos, en función de la estrategia, número de gateways y número de coronas utilizadas. Para el que se declaró como mejor estrategia se pasó al análisis de potencia para cada uno de los dos algoritmos de Dijkstra planteados en este trabajo.

Debido a la gran variedad de parámetros que podían modificarse, los resultados han estado condicionados a ciertas limitaciones.

Se ha trabajado únicamente con tres coronas, por lo que se limitan las posibilidades que podrían aparecer con una granularidad mayor, con lo que la potencia que se transmitida podría tener un abanico más amplio de valores.

Al limitar el número máximo de saltos a 3, puede que se estén desechando rutas con mayor número de saltos y menor coste y es por lo tanto, menor uso de potencia en las

transmisiones. En los casos con un número "alto"de coronas las transmisiones con más saltos son plausibles y se están eliminando.

Cuando se mide la potencia que se transmite, se están omitiendo otros valores dentro de la capa física que consumen potencia y podrían resultar relevantes. Por ejemplo las transiciones del estado de reposo a activo, o la potencia consumida en modulaciones, estado de recepción, etc.

Después de valorar estas limitaciones, y teniendo en cuenta el estudio realizado en el Capítulo 5 se puede llegar a la conclusión de que el método propuesto mejora las prestaciones en términos de potencia, frente a la operación tradicional. Cuanto mayor sea el número de coronas la mejora es mayor. Pero el aumento de coronas supone en cierto modo un incremento del número de saltos. En cualquier caso, como demuestran las simulaciones, si los escenarios son muy poblados el número de saltos necesario para conectarse a un gateway no suele superar los dos, por lo que esta limitación no sería totalmente restrictiva.

Utilizando despliegues sub-óptimos desde el punto de vista topológico se consiguen las mejores prestaciones ya que el despliegue de gateways es perfecto para que estén cerca al mayor número de nodos posibles y el número de saltos se reduce considerablemente.

6.2 Líneas Futuras

Para empezar, y como ya se dijo en el Capítulo 4, el estudio se realizó para el caso de una única interfaz, por lo que habría que adaptarlo para poder contar con múltiples interfaces, sobre todo el algoritmo de Dijkstra, ya que presenta ciertos retos en este campo.

Se ha utilizado un método de coronas, que mejora los resultados de eficiencia energética, aunque el caso óptimo sería poder transmitir a la potencia justa para recibir en el próximo salto la información sin pérdidas, es decir conocer la distancia del siguiente nodo y transmitir a la potencia adecuada a esa distancia. Con esto se podría conseguir el caso óptimo de transmisión.

El uso del Monitor es apropiado para este tipo de análisis, pero en redes reales no se tiene la capacidad de tener una entidad omnisciente que conozca toda la información de red. Es necesario emplear métodos de descubrimiento para encontrar los nodos adyacentes y el gateway más apropiado.

En este proyecto sólo se ha utilizado transmisión de un nodo hacia otro en todo el canal. Por lo que el estudio de posibles colisiones e interferencias queda pendiente de estudio.

Bibliografía

[1] L.S. Ferreira and L.M. Correia. Energy-efficient radio resource management in self-organised multi-radio wireless mesh networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*, pages 232 –236, sept. 2011.

[2] J.A. Irastorza, R. Aguero, and L. Muñoz. Manager selection over a hierarchical/distributed management architecture for personal networks. In *2nd International ICST Conference on Mobile Networks and Management, 2010. MONAMI 2010.*, 2010.

[3] Mauricio G. C. Resende and Renato F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10:59–88, January 2004.

[4] Ekram Hossain Teerawat Issariyakul. *Introduction to Network Simulator NS2*. Springer Science+Business Media, LLC, 2009.

[5] USC/ISI UC Berkeley, LBL and Xerox PARC. *The ns manual*. The VINT Proyect, 2010.

[6] Ramón Agüero Calvo and Jesús Pérez Campo. Adding multiple interface suport in ns-2. January 2007.