FACULTAD DE CIENCIAS

UNIVERSIDAD DE CANTABRIA Facultad de Ciencias

Proyecto Fin de Carrera

ALGORITMOS PARA EL ESTUDIO DE TÉCNICAS DE MULTI-PATH Y NETWORK-CODING EN REDES INALÁMBRICAS MULTI-SALTO

(Algorithmic study of multi-path and network coding techniques over wireless multi-hop networks)

Para acceder al Título de

INGENIERO EN INFORMÁTICA

Autor: Carlos Mª Rabadán Cebolla
Director: Ramón Agüero Calvo

Junio - 2013

Agradecimientos

En primer lugar, a mi director de proyecto Ramón Agüero, por confiar en mí al proponerme tan interesante y ambicioso trabajo, y por estar ahí siempre para resolver cualquiera de las infinitas dudas y problemas que le fui planteando. También destacar su gran labor previa como profesor durante la carrera. Su profesionalidad y comprensión motivaron mi interés por este campo de la ingeniería y su elección para basar en ello mi PFC.

A David Gómez, por ayudarme a participar en dos artículos de investigación y a socorrerme cuando más cuesta arriba se ponía el trabajo.

A la UC, por ofrecerme una beca de colaboración en el SdeI, en donde pude adquirir nuevas competencias a la vez que desarrollaba mi proyecto.

A la familia y a todos mis amigos, por estar siempre ahí. Por confiar en mí en todo momento, por difícil que se lo pusiera. Por distraerme cuando empezaba a obsesionarme con el trabajo y ayudarme a sustituir el ordenador por una cerveza.

A todos ellos, gracias.

Índice

Índice de Figuras

Resumen ejecutivo

A día de hoy existen multitud de técnicas y mecanismos para mejorar el rendimiento en redes de comunicación cableadas. Sin embargo, al trabajar con sistemas inalámbricos, son muchos los nuevos aspectos a tener en cuenta para alcanzar un comportamiento comparable, o al menos adecuado para satisfacer la cada vez mayor demanda de calidad por parte de los usuarios. En ese sentido, existen distintas posibilidades que actualmente se encuentran en fases de investigación, ya que necesitan de un nivel de maduración mayor para llegar a ser implementadas.

Se ha demostrado que dividir el tráfico a través de múltiples caminos (encaminamiento multi-camino o *multi-path*) incrementa el rendimiento de una red, en comparación con encaminamientos de una única ruta, en donde aparecen problemas como la fiabilidad o la congestión de la red. Sin embargo, las interferencias causadas por la naturaleza (intrínsecamente de difusión) de los medios inalámbricos pueden reducir notablemente los beneficios de tal aproximación.

Por otro lado, también se ha demostrado que permitir que los nodos intermedios de una red mezclen (codifiquen) la información procedente de distintas fuentes para enviarla hacia sus destinos correspondientes, donde será decodificada, puede aumentar su *throughput*. Esta técnica, que se conoce como codificación de red (*network-coding*), ha sido estudiada principalmente en entornos de red cableados, y resulta interesante analizar sus posibilidades en redes inalámbricas.

En este trabajo se ha hecho uso de las dos propuestas anteriores para implementar una serie de algoritmos en C++, basándose en la teoría de grafos. En primer lugar, un conjunto de funciones para hallar diferentes clases de caminos disjuntos en un grafo dado, con los que poder aplicar un tipo de encaminamiento *multipath*. En un segundo bloque, una serie de algoritmos para establecer la posibilidad de aplicar la técnica de *network-coding* en un escenario inalámbrico dado, así como para identificar los posibles nodos encargados de *mezclar* la información.

Todos los algoritmos implementados se validarán, mediante un exhaustivo proceso de análisis, utilizando despliegues de redes generados de manera aleatoria. Además, los resultados obtenidos serán llevados a un simulador de redes para comprobar, utilizando protocolos reales, cuáles son los posibles beneficios que pueden aportar.

Executive summary

Nowadays, there are several mechanisms to improve performance over traditional wired networks. On the other hand, some new challenges will spring while trying to apply the same techniques over wireless networks and obtain comparable performance gains. There are various possibilities that are currently being research by the scientific community, since they need a higher degree of maturity to go to real development and deployment.

It has been shown that splitting traffic over multiple paths (multi-path routing), can increase the network performance, as compared to legacy single-path routing, where there could be reliability and congestion issues. However, the interference caused by simultaneous transmissions, due to the intrinsic broadcast nature of wireless networks, can significantly reduce the benefits of such an approach.

On the other hand, it has been proved that allowing intermediate nodes mixing ("encoding") information from different sources in order to be sent joined to their respective destination nodes, where the packets will be decoded, can significantly increase the network throughput. This technique, known as "network-coding", has been already studied, but most of the existing works are done for wired network environments, and it becomes interesting to analyze its potential in wireless networks.

In this project we have used both of the previous proposals to implement a number of algorithms, using the C++ programming language, which are based on the traditional graph theory. On the first hand, we have implemented a set of functions to find a number of disjoint paths in a given graph, in order to apply multi-path routing. Afterwards, we also implemented algorithms to establish the feasibility of "network-coding" over wireless scenarios, to identify which nodes would need to take the coding role, mixing the information before transmitting it.

All implemented algorithms have been validated through an extensive simulation campaign, using randomly generated network topologies. Last, but not least, the obtained results will be exploited, over a network simulator, to assess, by means of real protocols, which are the potential benefits they might bring about.

1 Introducción

1.1 Descripción general del trabajo a analizar

El desarrollo de este proyecto está dividido, desde un principio, en dos partes bien diferenciadas, pero que tienen en común el entorno en el que se desarrollan: sistemas de redes inalámbricas de tipo multi-salto (*multi-hop*). Cada uno de estos bloques se corresponde con el tipo de técnica estudiada y sus correspondientes algoritmos implementados.

En primer lugar se estudiarán las de tipo multi-camino o *multipath*, cuyo objetivo consiste en encontrar y construir diferentes rutas independientes sobre redes inalámbricas. Para cada una de las funciones de este apartado, se comprobará su funcionalidad haciendo uso de distintos bancos de pruebas y se demostrarán las ganancias que aportan, en términos de *throughput*, comparando las diferentes técnicas entre sí.

En segundo lugar, se hará un análisis de la técnica conocida como *network-coding* para después explicar las funciones implementadas en el marco de este trabajo de manera empírica, basadas todas ellas en diferentes tipos de algoritmos de exploración y búsqueda de la teoría de grafos, con el objeto de diseñar una solución eficaz basada en esta técnica. Posteriormente, se expondrán una serie de ejemplos y pruebas realizadas para mostrar y comparar la funcionalidad del conjunto de funciones implementadas.

Para llevar a cabo el desarrollo de los algoritmos y funciones previamente mencionados, fue necesario un estudio en detalle de diversos aspectos de la teoría general de grafos y de algoritmos de autores como E. Dijkstra y Jin Y. Jen. De la misma forma, se implementaron una serie de funciones que permitiesen la creación de escenarios (grafos) aleatorios simples, donde el despliegue de los vértices fuese al azar, pero siempre cumpliendo una serie de requisitos, a especificar por el tipo de prueba en la que se fuesen a utilizar.

El lenguaje de programación elegido para todos los apartados realizados en este proyecto es C++, por razones de comodidad (familiaridad con el lenguaje y potencia del IDE escogido), y eficiencia computacional. Su naturaleza orientada a objetos y la existencia de múltiples tipos de estructuras de datos, permitirán un desarrollo modular de las diferentes clases y funciones implementadas. También fue necesario programar otros *scripts* en Matlab, como herramienta auxiliar para representar diferentes tipos de redes (en forma de grafos) y gráficas estadísticas basadas en los resultados obtenidos de los algoritmos.

1.2 Objetivos del PFC

Los objetivos que se persiguen en este trabajo también pueden dividirse en dos ramas distintas. En primer lugar, en el bloque correspondiente al multipath:

1. Analizar y aplicar diferentes aspectos de la teoría de grafos para estudiar y simular el comportamiento de tres algoritmos de encaminamiento multipath en un entorno de red inalámbrica multi-salto.

2. Elaborar las funciones necesarias y comprobar su correcto comportamiento para diferentes tipos de grafos como entrada.

3. Realizar un estudio mediante experimentos con escenarios generados aleatoriamente para observar la adaptabilidad de estas técnicas de encaminamiento. Se contabilizará también el número de rutas disjuntas halladas y su longitud. Representar gráficamente estos resultados mediante Matlab.

4. Adicionalmente se generarán un conjunto de resultados, analizando el nivel de *throughput* al aplicar las distintas funciones desarrolladas sobre despliegues aleatorios, mediante el uso del simulador ns-3, en colaboración con una línea de investigación abierta en el Grupo de Ingeniería Telemática de la UC.

En el bloque de network-coding se llevarían a cabo los siguientes pasos:

1. Elaborar un método experimental, en base a algoritmos de la teoría clásica de grafos, que permita hallar uno o varios candidatos a "nodo codificador" en un grafo dado, requisito fundamental a la hora de aplicar network-coding.

2. Llevar a cabo una serie de pruebas que demuestren el correcto funcionamiento del algoritmo implementado. Evaluar su capacidad de operación y su eficacia sobre un conjunto de escenarios de características particulares, y cuyos nodos ocupen posiciones alatorias.

1.3 Bloque de multipath

En este primer apartado del trabajo se realizará un análisis sobre los tres principales tipos de algoritmos de encaminamiento multipath: *link-disjoint*, *node-disjoint* y *zone-disjoint*, observando las diferentes propiedades de cada uno y su aplicabilidad.

Posteriormente se emplearían estos algoritmos sobre un conjunto de escenarios más complejos, como son las redes inalámbricas malladas, donde cada nodo de la red se comunicará con sus vecinos de manera inalámbrica dentro de un área de cobertura específica, facilitando las comunicaciones al reenviar paquetes de los nodos fuente a los destino mayoritariamente.

Para hacer posible la ejecución de estos algoritmos sobre un sistema de estas características será necesario elaborar una serie de funciones de creación y manipulación de

grafos, usando matrices de adyacencia como representación fundamental de los vértices y su conectividad.

Como se verá más adelante, los algoritmos anteriormente mencionados se basan en exploración de grafos y en la búsqueda de vértices, por lo que la implementación del algoritmo de Dijkstra y su uso como función auxiliar dentro de estos algoritmos será un elemento clave de esta sección.

Por último, se llevará a cabo una serie de bancos de pruebas de estos tres algoritmos sobre un número considerable de grafos, con diferentes características, para evaluar aspectos cuantificables, como pueden ser las ventajas de un algoritmo frente a otro en un mismo escenario o la cantidad de rutas independientes que es capaz de encontrar.

1.4 Bloque de network-coding

El objetivo principal de este segundo gran apartado del proyecto es algo más ambicioso que el anterior. Primero se hará una descripción detallada de la funcionalidad subyacente a la técnica conocida como *network-coding*, y como su aplicación puede afectar al rendimiento de una red. Asimismo, se describirá el estado actual de esta técnica en redes de comunicación cableadas, y de sus avances en sistemas inalámbricos como los vistos en el anterior apartado.

En segundo lugar se realizará un análisis del principal algoritmo implementado que solventa el problema de la localización del nodo, o conjunto de nodos, candidatos a "codificar" los paquetes de datos en escenarios con dos nodos fuente y dos nodos destino. De igual manera que en el apartado de multipath, también se trabajará con escenarios multi-salto, donde la comunicación inalámbrica y su naturaleza broadcast será un punto clave a la hora de analizar el comportamiento de este algoritmo.

Para mayor claridad a la hora de explicar el funcionamiento del esquema propuesto, previamente se llevará a cabo una breve descripción de los pasos dados para llegar a él. Conocer de antemano el funcionamiento de sus versiones anteriores y los distintos elementos que motivaron su evolución y cambios, facilitará en gran medida la comprensión del algoritmo definitivo.

Finalmente se realizará una serie de pruebas de este último algoritmo sobre un conjunto de escenarios aleatorios con el fin de evaluar, no solo su eficacia sobre grafos de propiedades pseudo-aleatorias, sino también hasta que punto es razonable su aplicación en términos más realistas.

1.5 Estructura de la memoria

Para continuar con el mismo orden establecido en este apartado introductorio, se hablará en un primer Capítulo 2 sobre el estado actual de las redes inalámbricas multisalto, así como de los diferentes protocolos de encaminamiento multipath. También se presentará la técnica de network-coding junto con una breve descripción de la misma, el estado del arte de los algoritmos de generación de grafos aleatorios, la importancia que tiene la "conectividad" en ellos y, por último, un rápido repaso a los algoritmos de exploración de grafos que serán empleados a lo largo del proyecto.

En el Capítulo 3 se presentarán los diferentes tipos de grafos aleatorios y semi-aleatorios y se describirá cómo fueron desarrollados, de cara a servir como plataforma de pruebas para el resto de apartados.

El Capítulo 4 se centra en el bloque de multipath, donde se analizarán los algoritmos implementados junto con las pruebas y estudios realizados con ellos, incluyendo la descripción de los experimentos realizados en cooperación con el GIT junto con el análisis de los resultados obtenidos en los mismos.

El Capítulo 5 por su parte, presenta el bloque de network-coding, con la descripción de todos los algoritmos implementados y el análisis de las pruebas y los resultados obtenidos con el más avanzado de ellos.

Por último, en el Capítulo 6 se recogen las principales conclusiones obtenidas de las secciones anteriores, junto con las posibles líneas de investigación que el trabajo realizado ha abierto.

2
Estado del arte

2.1 Redes Inalámbricas Malladas

Las redes inalámbricas malladas (*Wireless Mesh Networks* o **WMN**) son aquellas en las que existen uno o más nodos intermedios que envían y reciben paquetes para facilitar las comunicaciones entre nodos que no pueden hacerlo directamente. A diferencia de los sistemas más tradicionales en los que existe un enlace inalámbrico entre nodos, las MWN pueden ampliar la cobertura de una red, mejorar su conectividad, y permitir una mayor velocidad de transmisión de datos, lo que aumenta el rendimiento y proporciona un uso más eficiente del medio[1].

A la hora de trabajar con este tipo de redes, suelen representarse habitualmente como un grafo en el que los vértices simbolizan los nodos inalámbricos y las aristas representan un enlace entre dos nodos. Cada nodo posee un rango de cobertura inalámbrica determinado, y para que exista un enlace entre dos nodos ambos deben estar dentro del radio de cobertura del otro, como se muestra en la Figura 2.1. Este tipo de redes necesitan usar ciertas técnicas de enrutamiento que, al igual que sus homólogas cableadas, procuran elegir la mejor secuencia de nodos intermedios entre la fuente y el destino para retransmitir la información.

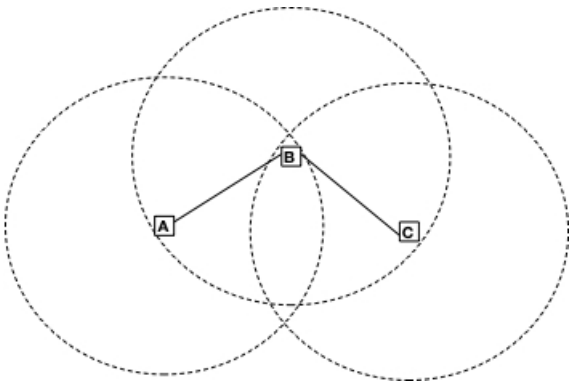

Figura 2.1: Representación de nodos inalámbricos en una MWN

2.2 Sobre la teoría de grafos

2.2.1. Algoritmos de exploración de grafos

A continuación se citan algunos de los algoritmos de búsqueda más importantes. Su uso como herramienta auxiliar ha resultado fundamental para muchas de las funciones implementadas, y frecuentemente ha sido necesaria su correspondiente adaptación con el objetivo de asegurar el correcto funcionamiento de los algoritmos desarrollados a lo largo del trabajo.

- **BFS** (Breadth-First Search): se trata de un conocido algoritmo de exploración de grafos, que comienza en un nodo *root* o raíz para después recorrer uno por uno todos sus "hijos". Se puede decir que explora la red "en anchura" puesto que primero ha de revisar todos los nodos de un nivel antes de avanzar al siguiente nivel. En la Figura 2.2 se observa un ejemplo de exploración de un grafo sencillo mediante BFS. Su eficiencia espacial usando matrices de adyacencia (como será el caso) es $\Theta(|V|)$, mientras que la temporal es $\Theta(|V| + |E|)$.
- **DFS** (Depth-First Search): de forma opuesta a la de BFS, DFS explora el grafo "en profundidad". Después de inspeccionar un vértice explora sus "hijos" de manera recursiva hasta llegar a uno que sea *hoja* (no tiene más descendientes), para después volver a ascender y seguir inspeccionando el resto. La Figura 2.3 muestra un ejemplo de este tipo de recorrido, con el orden de exploración representado por el número de cada vértice. Su complejidad espacial es $\Theta(|V|)$ y la temporal $\Theta(|E|)$ (exceptuando casos concretos).

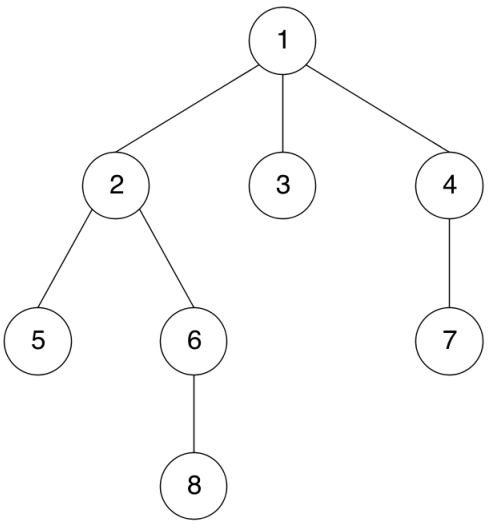

Figura 2.2: Recorrido en BFS

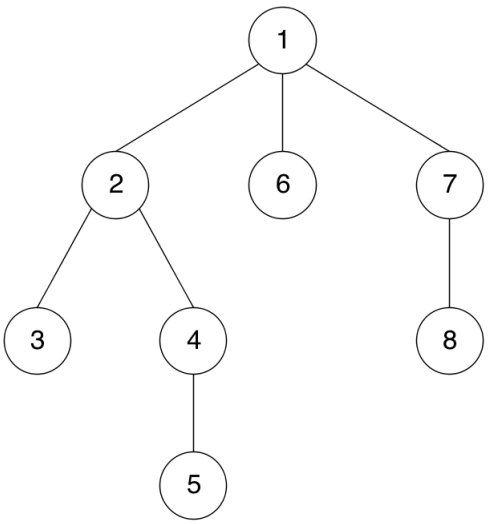

Figura 2.3: Recorrido en DFS

2.2.2. Algoritmos para hallar el camino más corto

Otra clase de algorítmica que se empleará muy frecuentemente en este proyecto es la dedicada a encontrar el *path* o camino más corto en un grafo dado. Esto es, la secuencia de nodos entre una fuente S y un destino D de menor número de saltos o de un coste global menor.

- **Dijkstra**: se trata de uno de los algoritmos más conocidos de la teoría de grafos, propuesto por Edsger Dijkstra en 1959[12]. Dado un grafo conexo y de distancias no negativas, encuentra el camino de coste mínimo entre un vértice dado y el resto de vértices del grafo. Su eficiencia computacional es $\Theta(|V|^2)$, y $\Theta((|E|+|V|)log(|V|))$ usando colas de prioridad.

- **KSP** (K-Shortest Paths): presentado por Jin Y. Yen en 1971[22], se trata de un algoritmo capaz de hallar los K caminos de menor coste entre dos nodos dados de un grafo, sin que estos contengan bucles. Su complejidad es de $\Theta(N^2 + KN)$ siendo N el número total de nodos del grafo y K la cantidad de caminos mínimos que se pretende hallar. El funcionamiento a grandes rasgos es el que se describe a continuación:

  1) Hallar el camino de menor coste (mediante Dijkstra por ejemplo) entre un nodo fuente s y su destino d.
  2) Aplicar una serie de modificaciones al camino previo para generar una ruta alternativa entre s y d, evitando crear ciclos internos. Almacenarla temporalmente.
  3) Repetir el paso 3 un número de veces equivalente a la longitud "en saltos" del último camino más corto hallado.
  4) Seleccionar de todos los caminos temporalmente almacenados el de menor distancia y guardarlo posteriormente.
  5) Repetir hasta K-1 veces

- **BFS** (Breadth-First Search): como se verá más adelante, este algoritmo puede usarse también para localizar el camino mínimo entre dos nodos puesto que recorre todos los vértices del grafo de forma progresiva. Aunque su comportamiento es análogo al del algoritmo de Dikstra, su eficiencia computacional es mayor, por lo que su uso resultará interesante en algunas de las funciones y algoritmos implementados.

2.2.3. La conectividad en los grafos

Este aspecto sobre los grafos cobrará una relevancia fundamental a lo largo de todo el proyecto, puesto que muchos de los algoritmos y funciones desarrolladas parten de un grafo conexo. Un grafo es conexo cuando existe un camino entre cualquier par de vértices o nodos. Al trabajar con grafos dirigidos, éstos pueden ser conexos o *fuertemente* conexos. Un grafo dirigido está *fuertemente* conectado cuando existe un camino entre cualquier

par de nodos. Nótese que un grafo *fuertemente* conexo es también conexo, y que uno no-dirigido conexo será siempre *fuertemente* conexo.

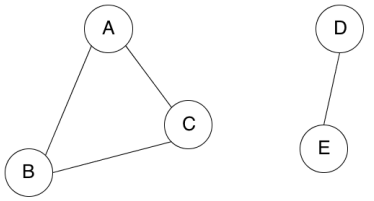

Figura 2.4: Grafo inconexo

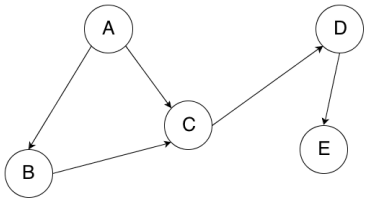

Figura 2.5: Grafo dirigido conexo

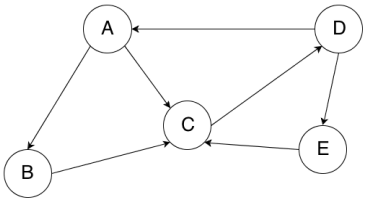

Figura 2.6: Grafo dirigido fuertemente conexo

2.2.4. Grafos aleatorios

Desde que Erdös y Rényi [10] lo utilizaran por primera vez, el interés por la teoría sobre redes y grafos aleatorios ha ido en aumento, y muchas de sus propiedades se han investigado detalladamente desde entonces. En este nuevo ámbito es importante disponer de un criterio por el cual saber si un grafo dado se puede calificar como uno aleatorio, y que por tanto cumpla ciertos requisitos necesarios para el problema concreto. Resulta interesante mencionar el *lema sobre regularidad de Szemerédi*, el cual postula que la existencia de cierta propiedad en un grafo aleatorio suele traducirse en una característica global de los grafos.

A priori, una red de estas características se obtiene partiendo de una serie de nodos n para posteriormente ir añadiendo de forma aleatoria enlaces entre ellos. Existen diferentes modelos de grafos aleatorios con sus respectivas distribuciones de probabilidad. Así surge la propuesta de E. Gilbert [11] como una de las más extendidas, donde cada arista o enlace e pose una probabilidad de existir p, y se expresa como $G(e, p)$.

La teoría de grafos aleatorios estudia las propiedades de este tipo de redes cuando se parte de distribuciones concretas. Una de las propiedades más interesantes es la probabilidad de que un grafo sea **conexo**. En concreto, la teoría de la *percolación* es la encargada de estudiar la conectividad en grafos aleatorios.

2.3 Técnicas multipath en redes inalámbricas multi-salto

El enrutamiento *multipath* permite el uso de múltiples caminos alternativos para la retransmisión de información dentro de una red. Se ha demostrado que el uso de diferentes técnicas multipath en WMN suponen una interesante alternativa al encaminamiento único, en términos de una mayor tolerancia a fallos y capacidad de recuperación ante errores, así como un mejor control de la congestión entre los nodos de la red al poder distribuir el tráfico entre distintos caminos[2][3].

Teniendo en cuenta la naturaleza compartida, inherente al medio inalámbrico de las WMN, surgen posibles problemas de rendimiento a la hora de realizar transmisiones

simultáneas de información. Nodos situados dentro del área de interferencia de aquellos que estén enviando o recibiendo datos en ese instante se ven obligados a detener su transmisión. El uso de soluciones basadas en técnicas multipath minimiza la interferencia entre nodos pertenecientes a caminos distintos y permite alcanzar una mayor tasa de rendimiento que la alternativa de enlace único.

Las ténicas multipath en este tipo de redes son principalmente dos: ***link-disjoint*** y ***node-disjoint***[2]. Dado un nodo fuente S y otro destino D, el conjunto de rutas *link-disjoint* está compuesto por aquellos caminos que no comparten ningún enlace o arista. De manera similar, el conjunto *node-disjoint* lo componen rutas sin un esquema común de nodos, a excepción de los propios extremos S y D. Por último, existe también un caso adicional denominado ***zone-disjoint***, cuyos caminos no pueden compartir enlaces, nodos, ni nodos adyacentes a los que conforman dicha ruta[14].

Natarajan Meghanathan analiza en su artículo [14] el comportamiento de diferentes protocolos de encaminamiento multipath mediante técnicas de teoría de grafos. Aunque el entorno en que lo hace es el de las redes tipo MANET (*Mobile Ad Hoc Network*), éstas no dejan de ser un caso particular de las WMN. En otro de sus trabajos([15]), Meghanathan analiza los algoritmos citados anteriormente y realiza una serie de pruebas para medir y comparar su rendimiento, de nuevo en redes MANET.

A lo largo del Capítulo 4 dedicado a multipath, se mencionará en varias ocasiones el protocolo **MPTCP**, que será utilizado para poner a prueba los algoritmos mencionados previamente. El protocolo MPTCP (*Multipath TCP*) es una evolución del TCP que explota la oportunidad que ofrece la existencia de múltiples interfaces en la mayoría de dispositivos de comunicación modernos.

Mediante el uso de esquemas multipath se pueden lograr mejoras notables en el rendimiento global del sistema, al poder dividir el tráfico enviado a través de distintas conexiones o "subflujos". Alguno de los objetivos que persigue este protocolo multi-camino son:

1. Un aumento del *throughput* en la red no inferior al alcanzado por el protocolo TCP tradicional en una ruta óptima.

2. No consumir más recursos de los que TCP necesitaría haciendo uso de un único camino.

3. Distribuir el tráfico de la red haciendo uso de subflujos menos saturados, y de esta manera evitar situaciones de congestión.

2.4 Network Coding

Los trabajos con la técnica de *network coding* comenzaron con un trabajo pionero de Rudolf Ahlswede [6] en el año 2000. En él demostró que se puede mejorar la capacidad de transmisiones *multicast* en una red cuando los nodos intermedios (*routers*) mezclan

la información de distintos mensajes en un solo paquete. Cuando el receptor recibe este paquete, podrá recuperar los datos que estaban destinados a él sólo si tiene a su disposición la información necesaria para decodificarlo.

Los procedimientos de network-coding tienen por tanto la responsabilidad de localizar "oportunidades de codificación", indicando qué paquetes pueden mezclarse y en qué punto de la red, pero siempre asegurándose de los receptores finales podrán decodificarlos. Para alcanzar la máxima capacidad en un entorno con tráfico multicast basta con usar codificación lineal, como ya demostraron Li et al. en [7].

Autores como Lun et al.[8] aplicaron esta técnica sobre sistemas inalámbricos, concretamente con antenas omnidireccionales, demostrando que minimizar los costes de comunicación se puede formular como un problema lineal y resolver de forma distribuida. En 2006, Katti et al.[19] presentaron su propuesta para redes inalámbricas, denominada COPE. Se trata de una nueva arquitectura de transmisión que, mediante la introducción de una entidad codificadora entre las capas IP y MAC, mejora sustancialmente el rendimiento en este tipo de redes. Dicha capa se encargaba de identificar las diferentes oportunidades de codificación y para retransmitir múltiples paquetes de datos en un sólo envío.

La mejor forma de explicar la mecánica de network coding es mediante el ejemplo de "la mariposa". La Figura 2.7 representa dicha topología, en la que el nodo S_1 quiere enviar el paquete p_1 al nodo D_1, mientras que S_2 pretende hacer lo propio con p_2 hacia el nodo D_2. Se supone que todos los enlaces tienen la capacidad de enviar un paquete por unidad de tiempo.

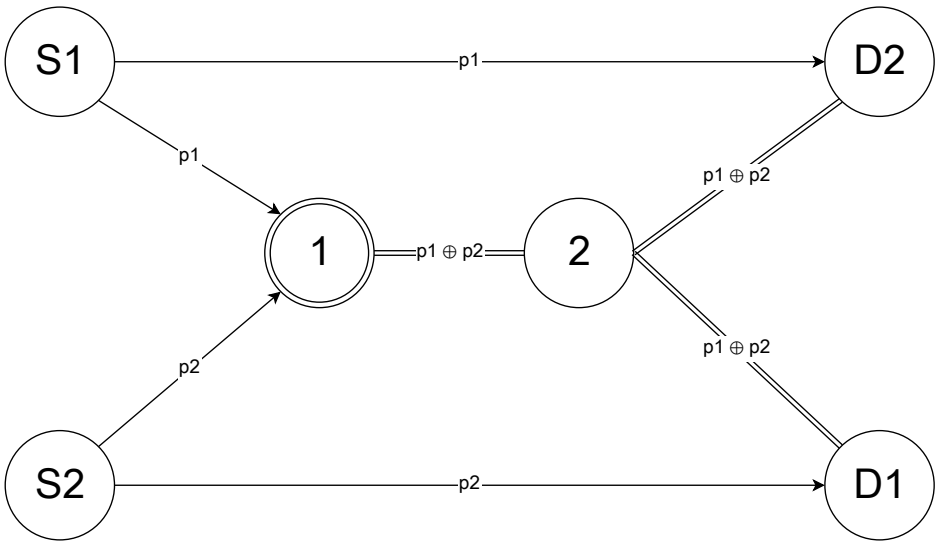

Figura 2.7: Grafo de la mariposa

Si los nodos intermedios 1 y 2 se limitan a reenviar los paquetes que les llegan, se formará un cuello de botella en el enlace situado entre ambos, que obligará a transmitir

primero uno de los paquetes y posteriormente el otro. En cambio, si el primero de los nodos intermedios *codifica* ambos paquetes (mediante la función XOR o cualquier otro método de combinación) y envía el mensaje $p_1 \oplus p_2$, llegarán a los nodos destino en el mismo instante. Este es un ejemplo simple de cómo, mediante network coding, es posible mejorar el rendimiento de una red en términos del número total de transmisiones y ocupación del medio.

3

Descripción de los algoritmos de grafos aleatorios implementados

3.1 Introducción

En este apartado se explicará de forma breve los diferentes algoritmos elaborados, su funcionalidad y su participación en las principales etapas del proyecto. Es necesario matizar que, aunque el objetivo final de este trabajo no es el estudio de este tipo de algoritmos, se consideró necesario idear e implementar una serie de funciones que den lugar a distintos tipos de escenarios aleatorios con los que trabajar posteriormente.

Todos ellos generan un grafo que, en mayor o menor medida, será aleatorio en cuanto a la posición de sus vértices o nodos. Lo que sí será diferente de unos a otros es la forma de calcular dichas posiciones, en forma de coordenadas espaciales dentro un plano euclediano de dos dimensiones (X, Y).

3.2 Análisis e implementación

Para una mejor representación jerárquica de los algoritmos implementados se irán exponiendo en orden de complejidad, comenzando por los más básicos y terminando por los más complejos.

3.2.1. Primer algoritmo: método de los dardos sencillo

Se trata del más sencillo e intuitivo de todas las funciones desarrolladas. Asimismo, será la más usada de todas, dado que su eficiencia radica en su sencillez. Dados tres parámetros de entrada:

- El *rango* de cobertura inalámbrica dependiente de la tecnología usada por los nodos de la red.[1]
- El *número de nodos* total a desplegar en el escenario y que habrán de ubicarse aleatoriamente.

[1] Aunque este algoritmo no haga uso del *rango* para generar el escenario, se trata de un valor fundamental del grafo para procesar las distancias entre sus nodos mediante otras funciones.

- El lado del *área* que conforma el espacio de trabajo, cuadrado en este caso.

La naturaleza de este algoritmo se asemeja bastante a la del lanzamiento de dardos en una diana, salvo que en vez de dardos son los vértices del grafo (con sus correspondientes coordenadas X e Y), y en lugar de una diana circular se trata de un área cuadrada. El algoritmo se asegura de que todos los "dardos" caigan siempre dentro del área establecida.

El principal inconveniente de esta función es que no asegura la conectividad del grafo. Esto conlleva que gran parte de los grafos que se generen mediante este algoritmo tengan que ser descartados, puesto que la mayoría de las funciones desarrolladas durante el proyecto requieren que el grafo con el que vayan a trabajar sea conexo.

3.2.2. Segundo algoritmo: método de los dardos conexo

El segundo algoritmo implementado hereda la mayor parte de la funcionalidad del primero, y además añade la capacidad de asegurar la conectividad del grafo resultante. Los parámetros de entrada siguen siendo los mismos (*rango*, *número de nodos*, *área*). Los pasos seguidos por esta función son los que siguen:

1. Iteración inicial: se sitúan todos los N nodos de forma aleatoria dentro del área cuadrada.
   a) Mediante el algoritmo de exploración BFS, se comienza un proceso de búsqueda desde el primer nodo, y se almacenan aquellos n nodos que son alcanzables, directa o indirectamente desde éste.
   b) Aquellos nodos que no se encuentren en el árbol generado por el BFS son descartados.
2. Resto de iteraciones: se sitúan aleatoriamente los $(N-n)$ nodos restantes en el área de trabajo.
   a) Se vuelve a repetir el recorrido BFS desde el primer nodo y se realiza la misma operación que en el paso 1.b con aquellos nodos que sean inalcanzables mediante este recorrido.
3. El algoritmo termina cuando los N nodos han sido situados dentro del área y todos ellos son alcanzables.

La función implementada, aunque logra obtener un grafo de N nodos perfectamente conexo, presenta una flaqueza importante que se puso de manifiesto en los primeros experimentos: todos los vértices del grafo tienden a situarse alrededor del vértice inicial (desde el cual se ejecuta el algoritmo BFS), con lo que al final se obtiene una distribución

más concentrada y poco natural de los nodos del grafo.

Es por ello por lo que se decidió descartar esta alternativa para las posteriores pruebas con diferentes clases de escenarios aleatorios en los bloques correspondientes a multipath y network-coding.

3.2.3. Tercer algoritmo: método secuencial

El tercero de los algoritmos implementados busca asegurar la ocupación de todo el área de trabajo longitudinalmente, esto es, ir situando los nodos de manera aleatoria de izquierda a derecha dentro del plano cuadrado. Para ello, los nodos se desplegarán según un avance aleatorio en el eje de abscisas, y su correspondiente desplazamiento proporcional en el eje de ordenadas (que puede ser positivo o negativo).

Los parámetros de entrada siguen siendo los mismos que en casos anteriores (*rango*, *número de nodos*, *área*). Los pasos a seguir de este proceso se enumeran a continuación:

1. El primero de los vértices se coloca de forma *manual* en el extremo izquierdo del área de trabajo ($X = 0$).
2. Cada uno de los $N - 1$ vértices del grafo restantes tendrá un avance aleatorio en el eje de abscisas, comprendido entre 0 y el *rango*.

   a) El avance en el eje de ordenadas podrá ser positivo o negativo de manera aleatoria. Será su magnitud la que dependerá directamente del avance en el de abscisas previamente obtenido, y se calculará haciendo uso del Teorema de Pitágoras[2].

   b) En caso de que el desplazamiento en el eje de ordenadas haga que la posición del vértice se salga del área de trabajo, éste se corregirá cambiando de signo dicho desplazamiento.

   $$Y = \pm y$$

   c) En caso de que se supere la longitud del área de trabajo en el eje de abscisas se hará uso de la función módulo sobre este valor:

   $$X = x \ mod \ area$$

3. El algoritmo finaliza cuando se hayan situado los N nodos dentro del área de trabajo.

Esta función adolece del mismo comportamiento pseudo-aleatorio que el caso anterior, puesto que los vértices tienden a situarse cerca del eje de abscisas dejando "despobladas" las áreas de los extremos del eje de ordenadas. Aunque por la forma que tiene de

[2] *En todo triángulo rectángulo el cuadrado de la hipotenusa es igual a la suma de los cuadrados de los catetos.*

ir desplegando los vértices en el plano de trabajo se trata de un algoritmo que siempre construye grafos conexos, también se descartó de cara a las pruebas finales.

3.3 Algunos ejemplos

Para una mejor apreciación de los resultados de estas y otras funciones a lo largo del proyecto, se programó una serie de funciones en Matlab para representarlos gráficamente. A partir de los parámetros básicos del grafo (número de nodos, radio de cobertura inalámbrica y dimensiones del área de trabajo) y un fichero con la descripción de los nodos junto con sus coordenadas, sería posible plasmar de forma gráfica la disposición de dichos nodos así como su conectividad en forma de enlaces. A continuación se muestra una serie de imágenes obtenidas de esta manera, pertenecientes a distintos grafos, pero con los mismos parámetros de entrada: $nodos = 16;\ rango = 20;\ area = 100$.

En el primer conjunto de imágenes (Figura 3.1) se puede observar la disposición de los nodos en el área de trabajo cuando se ejecuta el algoritmo de los *dardos*. A simple vista se aprecia que el grafo esta desconectado, por lo que se debería rechazar de cara futuras pruebas.

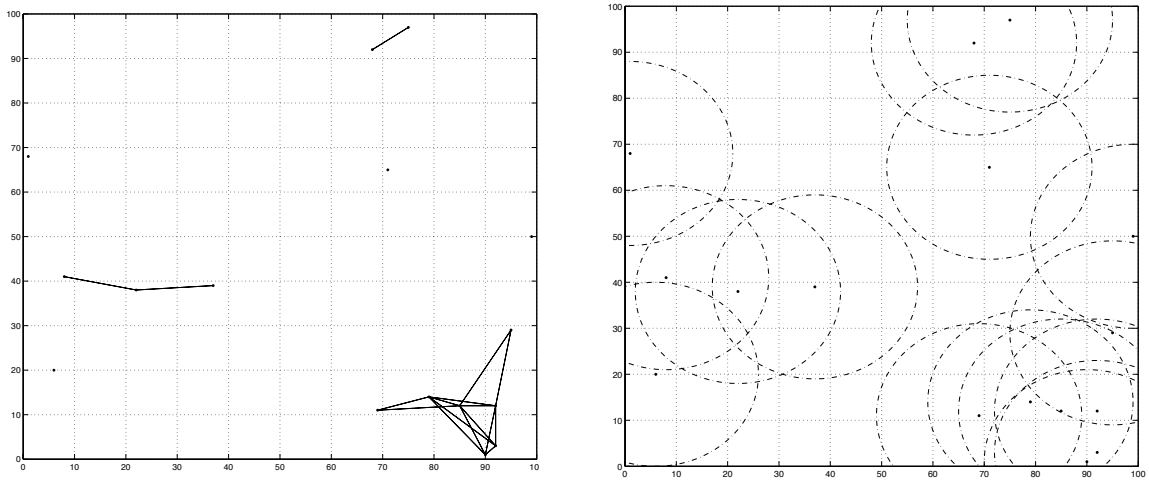

Figura 3.1: Grafo aleatorio inconexo.

El segundo grafo (Figura 3.2) refleja de manera clara la disposición artificial y "forzada" de los vértices que proporciona el segundo algoritmo implementado. Este tipo de despliegue permiten que el grafo sea conexo, pero a costa de concentrar todos sus nodos alrededor de un vértice central (*vértice cero*), ocupando solo una pequeña porción del área de trabajo.

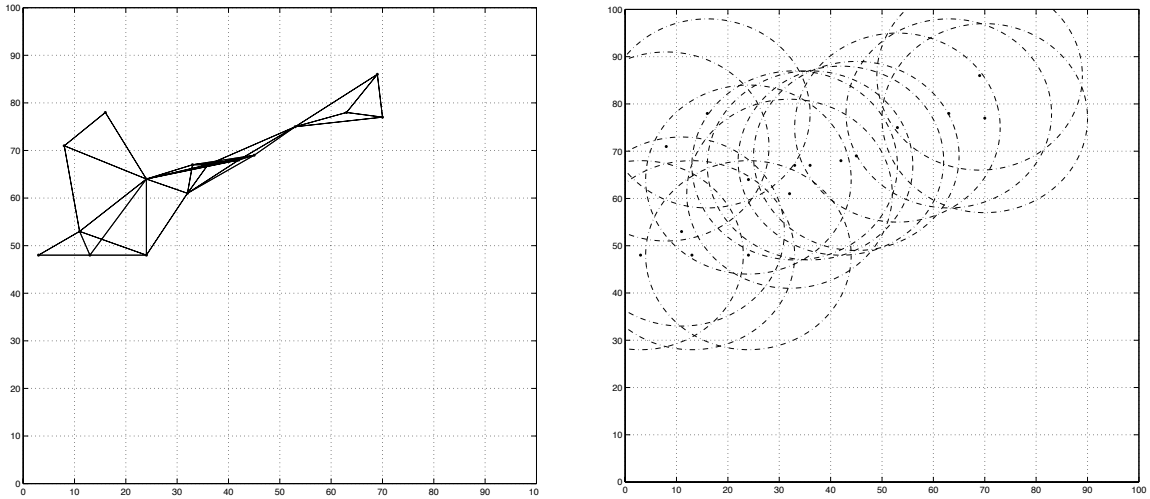

Figura 3.2: Grafo pseudo-aleatorio conexo.

En el último par de imágenes (Figura 3.3) se muestra lo que sería un grafo óptimo de cara a las pruebas: un despliegue completamente aleatorio (obtenido mediante el algoritmo de los *dardos*) y conexo, cuyos vértices se hallan relativamente bien repartidos en todo el escenario.

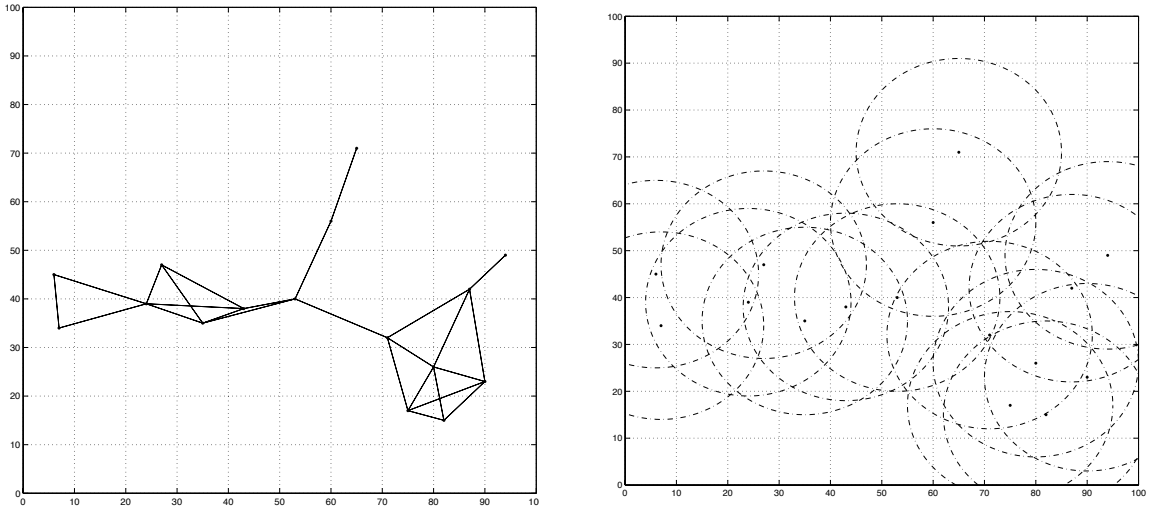

Figura 3.3: Grafo aleatorio conexo.

4

Descripción e implementación de los algoritmos multipath

4.1 Introducción

En este apartado se presentarán una serie de algoritmos que se han implementado para encontrar el conjunto óptimo de caminos disjuntos en redes inalámbricas multi-salto, con el objetivo de obtener las mejores prestaciones a la hora de transportar tráfico de una única conexión TCP de manera simultánea a través de múltiples caminos (o subflujos). Su desarrollo, como el resto de los principales módulos del trabajo, se llevó a cabo en C++, haciendo uso de diferentes tipos de estructuras de datos y otras funciones auxiliares más ligadas al apartado de la teoría de grafos.

Se seguirá utilizando el modelo de forma de disco dentro de un espacio euclídeo para representar los nodos en este tipo de redes inalámbricas. Para que exista un enlace entre los vértices del grafo ha de cumplirse la siguiente condición:

> "Dados dos nodos u y v ambos con un radio de cobertura r, existirá un enlace que los una si la distancia euclídea normalizada entre u y v dividida entre r es como máximo 1."[14]

Por tanto, la distancia relativa entre este par de nodos (magnitud que será usada reiteradamente en este y otros apartados) viene dada por la siguiente expresión:

$$d(u,v) = \frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{r}$$

Donde x_1 e y_1 son las coordenadas cartesianas del primero de los vértices y x_2 e y_2 las del segundo. Por último, es necesario destacar que cada uno de los caminos disjuntos que se hallarán mediante los tres algoritmos anteriormente mencionados serán siempre los más cortos, principalmente en términos de distancia en *número de saltos* entre el nodo fuente y destino. Para ello se hará uso de una variante del algoritmo clásico de Dijkstra que encuentre, únicamente, la ruta mínima entre dos nodos dados.

En la última parte de este capítulo se hablará sobre las pruebas realizadas con los algoritmos implementados, como parte fundamental de una serie de simulaciones llevadas a cabo con la herramienta **ns-3**[16] para medir el rendimiento del protocolo MPTCP.

4.2 Implementación

4.2.1. Link-disjoint

El primero de los algoritmos multipath estudiados es el denominado *link-disjoint*, que consiste en hallar el conjunto P_L de caminos entre dos nodos s y d dentro de un grafo $G(E,V)$ dado, que sean disjuntos en cuanto a los enlaces que unen cada uno de los nodos intermedios. El procedimiento seguido para hallar P_L es el siguiente:

1. Hallar el camino mínimo p_{sd} mediante Dijkstra y almacenarlo en P_L.
2. Eliminar todos los enlaces que formaban parte de p_{sd} del grafo original para obtener un grafo modificado $G_L(V, E^L)$ con las aristas restantes.
3. Repetir los pasos *1* y *2* hasta que no sea posible obtener más caminos entre s y d.

Al finalizar el procedimiento anterior, P_L estará formado por todos los posibles caminos *link-disjoint* del grafo entre los nodos s y d. La Figura 4.1 muestra la evolución de un grafo a medida que se van eliminando los enlaces de los caminos seleccionados.

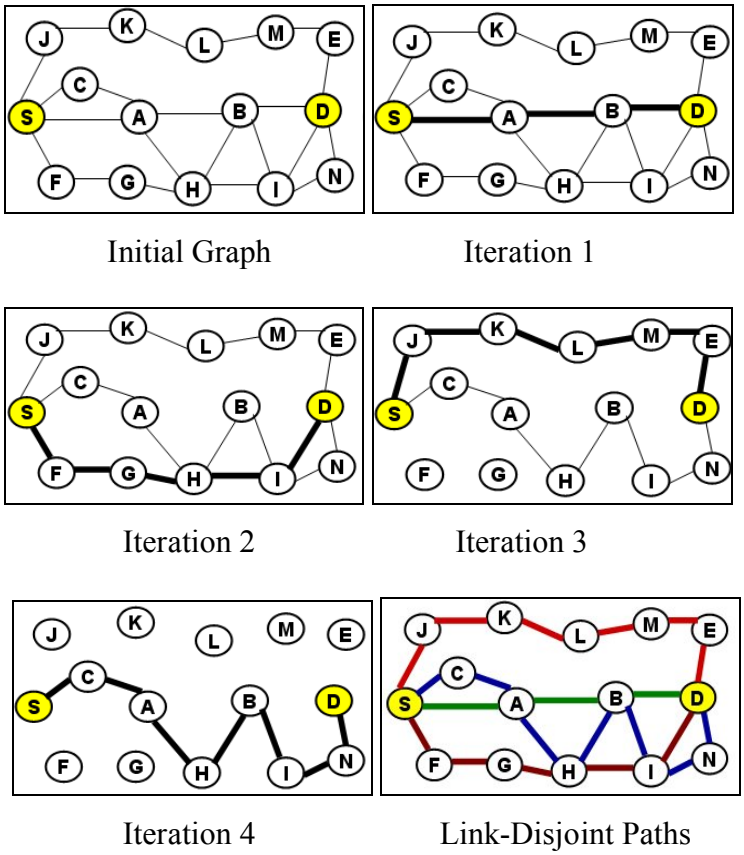

Figura 4.1: Evolución del grafo a medida que se aplica el algoritmo *link-disjoint*

4.2.2. Node-disjoint

El segundo algoritmo que se analizará es el conocido como *node-disjoint*. El procedimiento a seguir es muy similar al del anterior, salvo que en esta ocasión se eliminarán tanto los enlaces intermedios como los nodos que formen parte de los caminos hallados previamente. En la figura 4.2 se puede apreciar el funcionamiento de este algoritmo sobre el mismo grafo y, como era de esperar, se obtiene un conjunto P_N de caminos disjuntos de menor tamaño que el del algoritmo *link-disjoint*, por la naturaleza más restrictiva del esquema actual.

A continuación se citan los pasos a seguir en el procedimiento de *node-disjoint* para obtener P_N:

1. Calcular el camino más corto p_{sd} mediante Dijkstra y almacenarlo en P_N.
2. Eliminar todos los enlaces y nodos que formaban parte de p_{sd} del grafo original, a excepción de los nodos extremos s y d. De esta manera se obtendrá un grafo modificado $G_N(V, E^N)$ con las aristas y vértices restantes.
3. Repetir los pasos *1* y *2* hasta que no se puedan obtener más caminos mínimos.

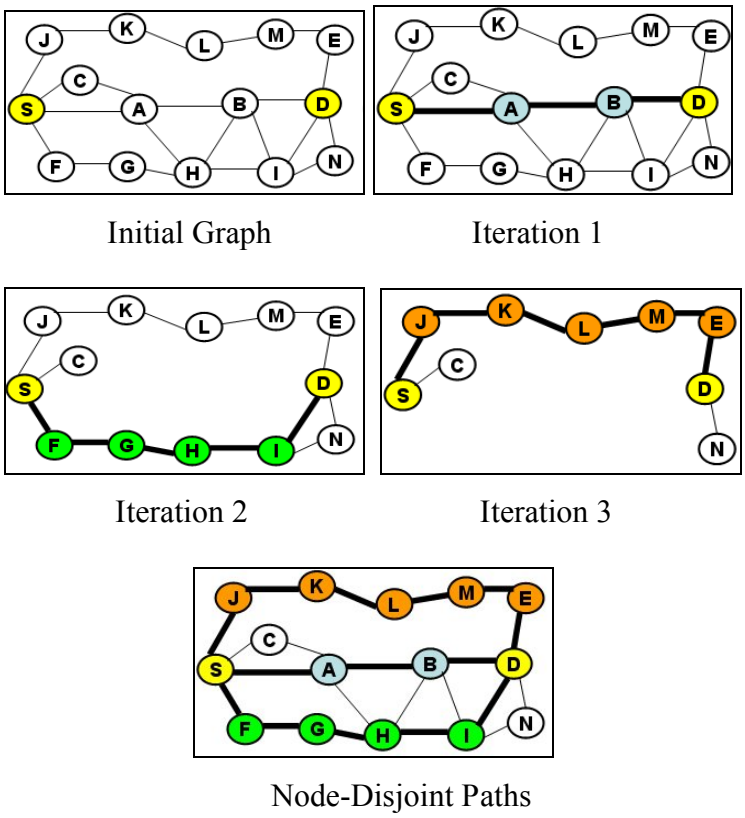

Figura 4.2: Evolución del grafo a medida que se aplica el algoritmo *node-disjoint*

4.2.3. Zone-disjoint

El último algoritmo multipath que se verá es el llamado *zone-disjoint*. Consiste en ir eliminando del grafo original no solo los enlaces y nodos pertenecientes al último camino encontrado, sino también sus nodos adyacentes. La Figura 4.3 representa el comportamiento de este algoritmo sobre el mismo grafo de los casos anteriores. Es necesario matizar que, ni los nodos adyacentes a s y d no petenecientes al camino p_{sd} ni los propios nodos extremos deben eliminarse durante la ejecución del procedimiento.

Los pasos a seguir para ejecutar este algoritmo son los siguientes:

1. Calcular el camino más corto p_{sd} mediante Dijkstra y almacenarlo en P_Z.
2. Eliminar todos los enlaces y nodos que formaban parte de p_{sd} del grafo original a excepción de los nodos extremos s y d. Eliminar también aquellos nodos adyacentes a los pertenecientes a p_{sd}, exceptuando s, d, y los vecinos de estos. Con ello se obtendrá un grafo modificado $G_Z(V, E^Z)$ con los nodos y enlaces restantes.
3. Repetir los pasos *1* y *2* hasta que no se puedan obtener más caminos mínimos.

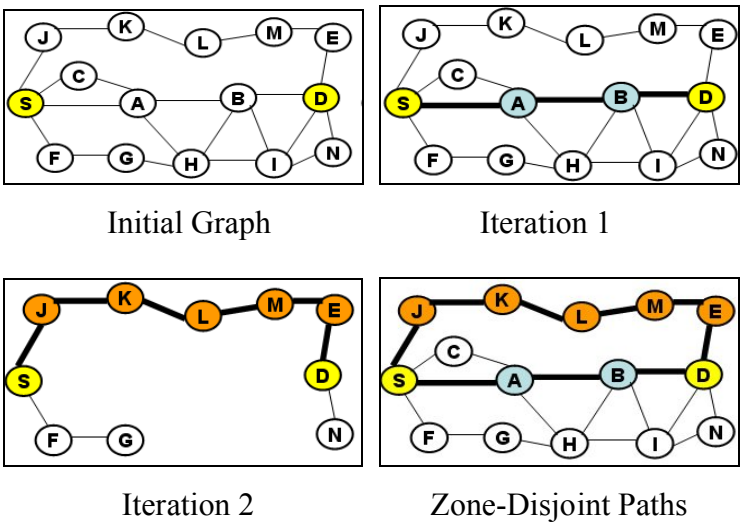

Figura 4.3: Evolución del grafo a medida que se aplica el algoritmo *zone-disjoint*

Se trata del algoritmo más restrictivo de los tres, dado que elimina a todos los nodos vecinos de los pertenecientes al último camino. De hecho, Meghanathan [15], al comparar el rendimiento de los algoritmos *link* y *node-disjoint* frente al de *zone* destaca la capacidad de este último para forzar que las rutas que encuentre tengan un factor de acoplamiento cero, mientras que en los otros dos es probable que los nodos acaben en contienda por acceder al medio compartido.

Al observar el caso de *node-disjoint*, se descartan los nodos que forman parte del camino *s-d* pero no tiene en cuenta a sus vecinos, que en redes inalámbricas podrían

provocar el denominado efecto de "terminal oculto" . Cuando uno de los nodos intermedios, que pertenece a una de las rutas halladas por el citado algoritmo, intenta acceder al medio inalámbrico para comunicarse con el siguiente nodo es probable que haya un vecino en el área de cobertura que pueda interferir en las transmisiones de aquel, al estar éste intentando transmitir también.

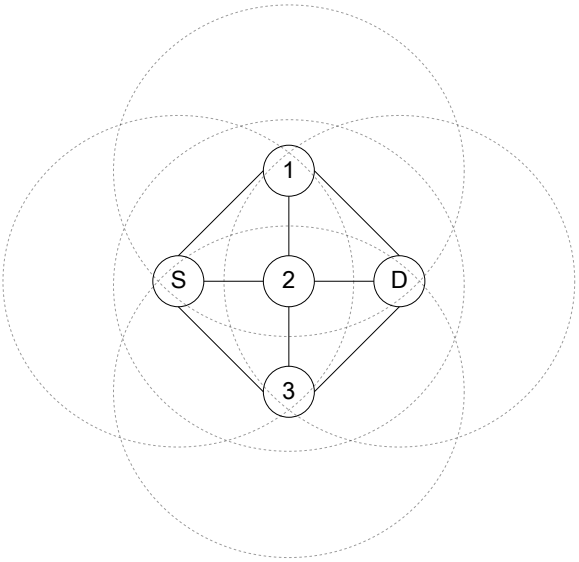

Figura 4.4: Esquema de terminal-oculto.

En la Figura 4.4 puede observarse un escenario donde el nodo 2 puede ocasionar el problema del terminal oculto para los otros nodos intermedios 1 y 3, siendo S y D los nodos fuente y destino respectivamente. Tanto si se utiliza LD como ND, las rutas obtenidas por el algoritmo serán tres: (S,1,D), (S,2,D) y (S,3,D). Como el área de cobertura inalámbrica del nodo 2 alcanza al resto, podrá interferir en las comunicaciones de 1 y 2. Este es un comportamiento que el algoritmo ZD evitaría, ya que descartaría el nodo 2 para hallar los caminos disjuntos (S,1,D) y (S,3,D).

4.3 Pruebas realizadas

En este apartado se describen los diferentes "bancos de pruebas" realizados para analizar los algoritmos explicados previamente en conjunción con las funciones generadoras de grafos aleatorios. El objetivo es probar la funcionalidad de los algoritmos multipath y observar los distintos resultados que se obtendrían al ir alterando los parámetros base de los grafos en los que se ejecutasen. Estos grafos (o escenarios) serán generados de manera aleatoria y masiva, y tendrán características particulares dependiendo del tipo de prueba efectuada.

Otros resultados que se estudiarían son el porcentaje de los escenarios en los que se podría establecer satisfactoriamente cada uno de los tipos de enrutamiento multipath

vistos. El número total de rutas halladas, así como la longitud media de éstas, también serían factores estudiar.

La funcionalidad de estos algoritmos, así como los escenarios aleatorios sobre los que se aplicarían los distintos experimentos, serán usados como base para un conjunto de simulaciones llevadas a cabo en el marco de una investigación abierta en el Grupo de Ingeniería Telemática (GIT) de la UC, mediante el ya mencionado simulador ns-3. El objetivo final consiste en demostrar un aumento del rendimiento conseguido por el protocolo MPTCP en redes inalámbricas malladas (*Wireless Mesh Networks*), sobre las soluciones tradicionales de camino único que ofrece el protocolo TCP estándar.

4.3.1. Implementación de un *parser*

Para que el simulador fuese capaz de procesar los datos de los escenarios en los que se llevarían a cabo las pruebas, así como los resultados de los algoritmos previamente descritos al aplicarse sobre estos escenarios, fue necesario implementar un *parser*. La función de esta herramienta es triple:

a) Analizar y procesar la lista de nodos del grafo dado, con su índice, sus coordenadas y diferenciando la fuente y el destino del resto de nodos intermedios.

b) Procesar la matriz de distancias entre nodos para generar un fichero, también en forma de matriz, basado en el modelo de pérdidas por propagación en el que se indique en cada fila/columna la calidad del enlace entre los dos nodos correspondientes.

c) Analizar la configuración del grafo dado para construir y exportar la tabla de rutas de la red en un fichero adicional. Para cada uno de los tres algoritmos estudiados habría un fichero de esta clase, conteniendo cada uno los caminos disjuntos obtenidos con los diferentes algoritmos.

Por último, mencionar también la utilización del programa Matlab para la interpretación y representación gráfica de muchos de estos escenarios, así como las diferentes gráficas estadísticas sobre las rutas halladas que se generarían a lo largo de los experimentos.

Una función auxiliar del *parser* sería la encargada de exportar diferentes clases de valores medios sobre los bancos de pruebas con grafos aleatorios. La información estadística a recopilar para futuros análisis sería:

a) Llevar la cuenta del número de escenarios (conexos) en los que es posible utilizar técnicas multipath.

b) Contabilizar el número de rutas halladas por los distintos algoritmos multi-camino.

c) Medir la longitud en número de saltos de los dos caminos más cortos proporcionados por los algoritmos. El primero siempre sería el mismo para los tres, ya que todos utilizan el algoritmo de Dijkstra para calcularlo, mientras que el segundo ya dependería del algoritmo multipath usado.

4.3.2. Tipos de escenario

A la hora de generar los distintos despliegues aleatorios para las pruebas, se propusieron dos tipos de escenario, ambos con nodos estáticos (una vez generado el grafo, sus vértices no cambiarán de posición):

I. El primero sería el clásico ejemplo de los *dardos*, donde todos los nodos estarían situados de manera aleatoria, pero siempre cumpliendo que el grafo sea conexo (existe, al menos, una ruta entre cualquier par de nodos).

II. Similar al anterior, pero 2 de los nodos tendrían siempre su posición configurada de antemano, manteniéndola fija durante las pruebas con este tipo de escenario.

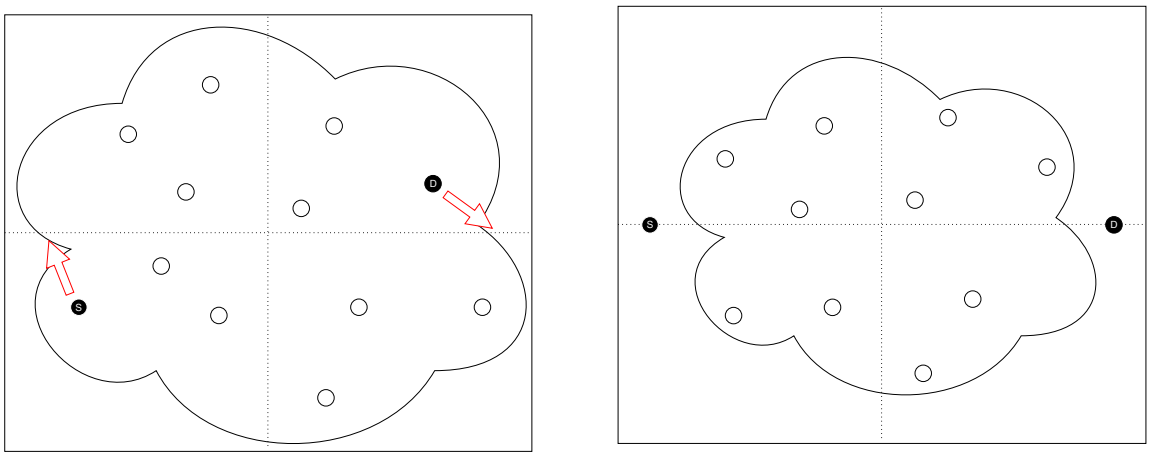

Figura 4.5: Tipos de escenarios generados.

Elegir qué vértices tomarían los roles de nodo fuente y destino era un elemento clave de cara a las pruebas llevadas a cabo. En un principio se pensó usar el modelo de escenario II, con los nodos fuente y destino situados siempre de forma fija en unas coordenadas seleccionadas de antemano y jugar con las posiciones del resto de nodos intermedios, pero finalmente se optó por usar el modelo I.

Con una organización de vértices completamente aleatoria[1] como la del primer modelo de escenario se pretendía lograr un mayor grado de validez estadística en los experimentos. En este caso los nodos fuente y destino serían asignados a aquellos vértices que estuviesen más cerca de una pareja de coordenadas elegidas previamente. Por defecto, estas coordenadas serían (20,50) y (80,50), de forma que los extremos de la comunicación estuvieran siempre más cerca de los extremos y permitir así que los otros nodos se situaran en regiones más centrales.

[1]Partiendo de la base teórica de que es imposible lograr la "auténtica" aleatoriedad con los medios de los que se dispone.

4.3.3. Esquema de las pruebas

El *benchmark* que se usó para las pruebas con los diferentes algoritmos multipath implementados consta de diferentes partes, de forma que al terminar de ejecutar se obtendría toda la información necesaria sobre los escenarios evaluados y los resultados de dichos algoritmos. La serie de pasos que sigue este banco de pruebas es la siguiente:

1) Genera un grafo completamente aleatorio a partir de una serie de parámetros concretos (número de vértices total, rango de cobertura de los nodos y dimensiones del área de trabajo).

   1.1) En caso de no ser conexo, el grafo es descartado y se repite la operación.

   1.2) Aplica el algoritmo multipath de *zone-disjoint* sobre el grafo conexo[2]. En caso de no encontrar al menos dos rutas independientes descarta este escenario y se repite el primer paso.

2) Aplica el resto de algoritmos multipath sobre el grafo actual, almacenando temporalmente las características obtenidas de los caminos hallados por los tres algoritmos.

3) Exporta el grafo actual en un formato adecuado para el simulador ns-3, usando la función correspondiente del *parser*.

4) Repite los anteriores pasos un número determinado de iteraciones para obtener unos valores estadísticos lo suficientemente fiables.

5) Exporta los valores estadísticos obtenidos de las funciones multipath. Este último conjunto de datos será empleado por el simulador ns-3, sino que serán procesados por Matlab para producir las gráficas correspondientes.

Es necesario destacar que todos los escenarios que participaron en las pruebas poseían siempre capacidad multi-camino, es decir, en todos ellos es posible encontrar al menos una ruta alternativa al camino más corto entre los nodos fuente y destino seleccionados. Ello no quita, por otro lado, que fuese posible calcular cuántos de los escenarios generados tuviesen que ser descartados precisamente por no cumplir este requisito. Del conjunto global de escenarios generados, sólo cerca del 5 % de éstos cumplieron los requisitos exigidos para las pruebas llevadas a cabo.

Garantizar ese aspecto de los escenarios era imperativo, puesto que el objetivo era reunir un número suficiente de escenarios donde fuera posible, mediante el protocolo MPTCP, dividir el tráfico de la red entre los diferentes caminos hallados previamente con los algoritmos implementados. El último paso de las pruebas sería medir y comparar los resultados de rendimiento de esta propuesta frente a la solución clásica *unipath* que ofrece TCP.

[2]La razón por la cual se aplica primero este algoritmo es porque es el más restrictivo de los tres. Si se halla al menos dos caminos con ZD es seguro que existirán dos o más con el resto de algoritmos.

La Figura 4.6 muestra un ejemplo ilustrativo de un escenario generado aleatoriamente que formó parte del banco de pruebas multipath. Se trata concretamente de un grafo de 16 vértices con un rango de cobertura de 25 situado en un área de trabajo de 100x100. Cumple todas las propiedades exigidas en las pruebas puesto que es conexo (no tiene ningún vértice aislado del resto) y posee capacidades multi-camino.

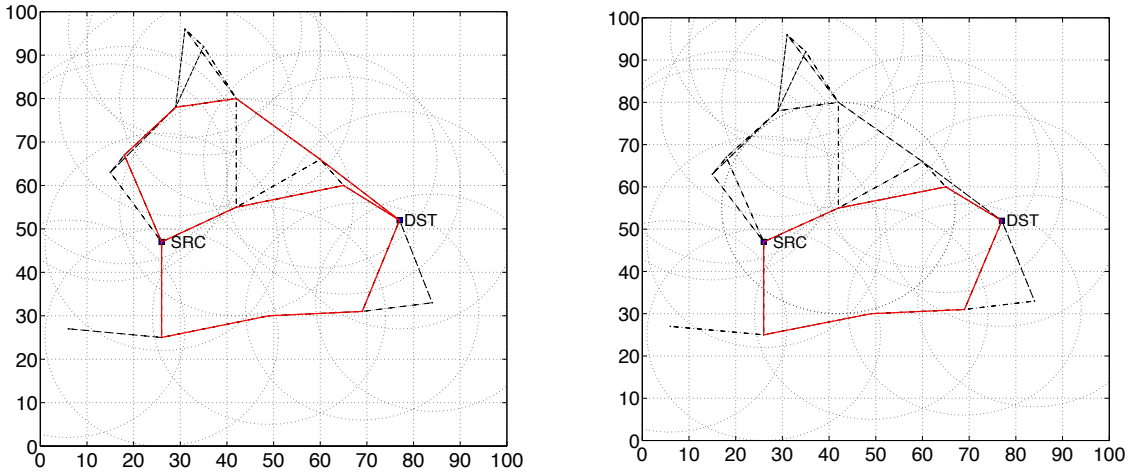

Figura 4.6: Algoritmos multipath aplicados sobre un escenario aleatorio.

En la imagen de la izquierda pueden observarse, en rojo, los tres caminos que tanto el algoritmo LC como ND encontraron entre los vértices destacados *SRC* y *DST*. La primera ruta y la más corta, con 3 saltos de distancia, es la central. Situada por debajo de ésta, la segunda ruta tiene una longitud de 4 saltos entre fuente y destino. La última de las tres, ubicada en la parte superior del grafo, posee una distancia de 5 saltos hasta llegar al vértice destino y al igual que las otras, es disjunta en enlaces y nodos.

La imagen de la derecha muestra el mismo escenario, destacado en rojo los dos únicos caminos que encuentra el algoritmo ZD. Si se observa con detenimiento se ve que, el primer nodo de la ruta central, alcanza con su radio de cobertura inalámbrica (representado como una circunferencia de puntos) los vértices situados justo encima del nodo fuente, con lo que quedarían descartados para posibles rutas alternativas. De hecho, se ha observado durante el conjunto de las pruebas realizadas que la probabilidad de que el algoritmo ZD encuentre más de dos caminos disjuntos es prácticamente nula.

4.3.4. Resultados

Para mayor claridad se dividirá esta sección en dos partes. La primera correspondiente a los datos ofrecidos por los tres algoritmos multi-camino implementados, y la segunda referida a los resultados de rendimiento obtenidos por el protocolo MPTCP sobre dos escenarios, con una y dos interfaces, frente al de TCP haciendo uso de esos mismo

escenarios aleatorios y enrutamientos multi-camino.

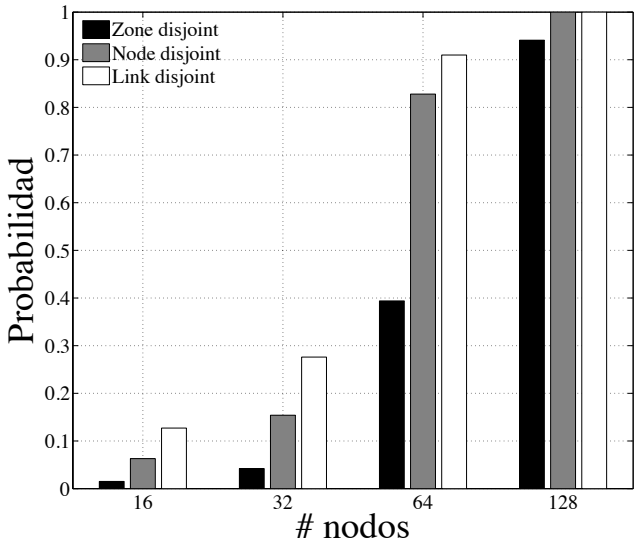

Figura 4.7: Porcentaje de escenarios con capacidad multi-camino.

El primero de los resultados que se estudiaron era el correspondiente al porcentaje de escenarios en los que era posible aplicar los diferentes algoritmos multipath implementados. La Figura 4.7 muestra esta estadística obtenida sobre varios conjuntos de 1000 escenarios, formados respectivamente por 16, 32, 64 y 128 nodos. Recordar que, para que se considere factible usar un protocolo multicamino entre dos nodos de una red específica, ha de existir, por lo menos, dos caminos independientes entre dichos nodos.

Se observan unos resultados que, aunque previsibles, no dejan de ser interesantes porque muestran la naturaleza de los tres algoritmos sobre diferentes escenarios aleatorios. Para configuraciones con menor número de nodos, la probabilidad de poder aplicar estos algoritmos multi-camino es muy inferior a cuando el escenario dispone del doble o el cuádruple de nodos con los que generar rutas disjuntas.

También salta a la vista el comportamiento mucho más restrictivo del último de los algoritmos implementados (*zone-disjoint*) frente a los otros dos, por ejemplo en escenarios con 64 nodos, la posibilidad de usar multipath con *link* y *node-disjoint* es del 80-90 % respectivamente, cuando el *zone* apenas alcanza un 40 % de los casos.

Otro de los aspectos a tener en cuenta de los experimentos era el número total de rutas halladas por cada algoritmo. Los datos obtenidos indican que, mientras que LD se mueve en valores de 2 a 4 llegando incluso a alcanzar 6 caminos independientes, ND a penas pasa de los 3 en la mayoría de los casos. En ZD no se obtuvo más de 2 caminos disjuntos en ninguno de los escenarios lo que no quiere decir, como se mencionó anteriormente, que exista alguna posibilidad más.

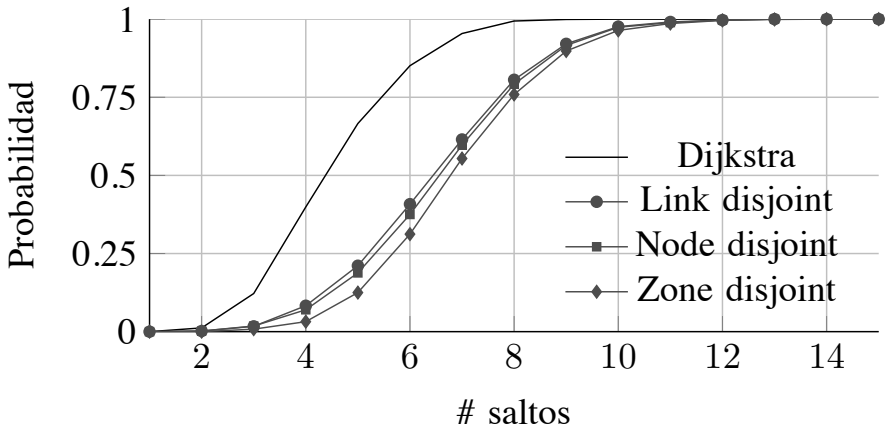

Figura 4.8: Cdf del número de saltos de las dos rutas preferentes.

El último de los parámetros que se estudió fue la longitud (en número de saltos entre nodos) del primer y segundo camino más corto hallados por los tres algoritmos. La Figura 4.8 es una representación de la función de distribución acumulada (cdf) de esta estadística. Por un lado, está la curva que marca el camino más corto (el de Dijkstra) que será común para los tres algoritmos, mientras que por otro lado, están las otras curvas de mayor cantidad de saltos todas ellas. Se observa una vez más como el algoritmo ZD consigue los peores resultados al obtener un recuento de saltos ligeramente superior en la segunda ruta.

En cuanto a los resultados que se obtuvieron en las pruebas con el simulador ns-3 y el protocolo MPTCP, es necesario subrayar primero algunos detalles antes de mostrar los resultados correspondientes. Se usarán escenarios aleatorios similares a los empleados en las pruebas de los algoritmos multipath, con la diferencia de que solo se utilizarán aquellos en los que es posible aplicar el algoritmo ZD (y por lo tanto también LD y ND) con un mínimo de dos rutas disjuntas. En la simulaciones, y con el fin de comparar los resultados obtenidos, se utilizarán tres configuraciones distintas:

1. TCP, con un único flujo a través del camino más corto (aquel que se calculó en primera instancia mediante Dijkstra y que es común a los tres algoritmos multipath).

2. MPTCP sobre un único interfaz, que deberá disponer de dos direcciones IP. Su rendimiento se verá afectado en gran medida, ya que los subflujos comparten el mismo canal inalámbrico, con lo que el número de nodos intentando acceder al medio simultáneamente y el de colisiones serán más elevados.

3. MPTCP sobre dos interfaces distintas, con direcciones IP independientes con lo no habrá ninguna clase de interferencia entre los subflujos.

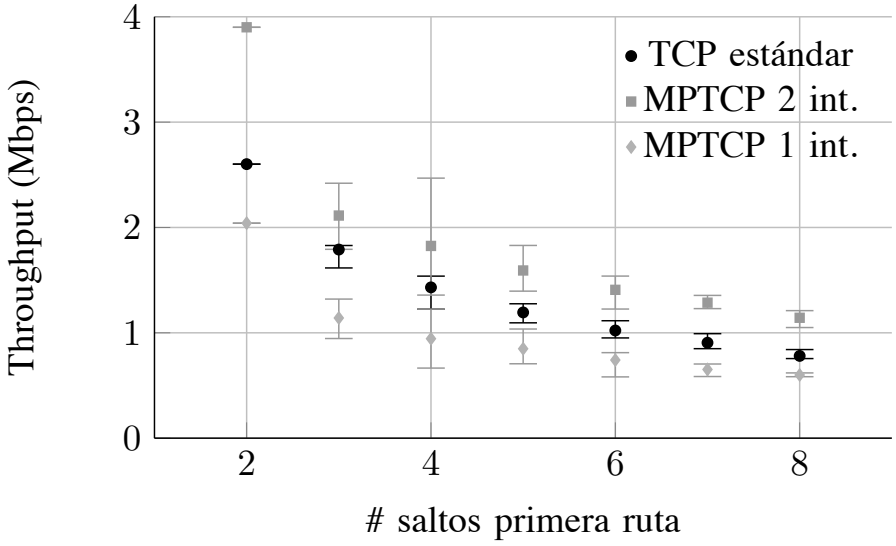

Figura 4.9: Throughput medio en función del número de saltos del camino más corto.

El primero de los resultados analizados de las simulaciones será el *throughput* medio en Mbps para cada una de las tres soluciones citadas previamente. La Figura 4.9 muestra el rendimiento en función de distintos valores de longitud de la primera ruta hallada (la más corta).

Dos aspectos relevantes a destacar: el primero es que la solución basada en el protocolo MPTCP con 2 interfaces vence claramente a las otras dos, llegando a alcanzar tasas hasta un 50 % mayores en el rendimiento agregado, a excepción de situaciones puntuales donde la longitud de la segunda ruta más corta es mucho mayor que la primera. La segunda conclusión que se deduce de los resultados es el bajo rendimiento que ofrece el protocolo MPTCP con una única interfaz. El elevado número de nodos intentando acceder al canal, produciendo de esta manera una gran cantidad de colisiones, alarga los tiempos de espera y obteniene un rendimiento muy inferior al visto en TCP.

El segundo resultado estudiado en las simulaciones es la diferencia de rendimiento obtenido al aplicar las rutas devueltas por los tres algoritmos multipath implementados. Se encontró que, en la mayoría de las ocasiones, era complicado hallar dos rutas que fuesen disjuntas al aplicar ZD (como se observa en la Figura 4.7). Además, resultaba muy común que la longitud en saltos de la segunda ruta más corta hallada por este algoritmo fuese en proporción mayor que la primera, lo que penalizaba el ancho de banda agregado en una conexión mediante el protocolo MPTCP.

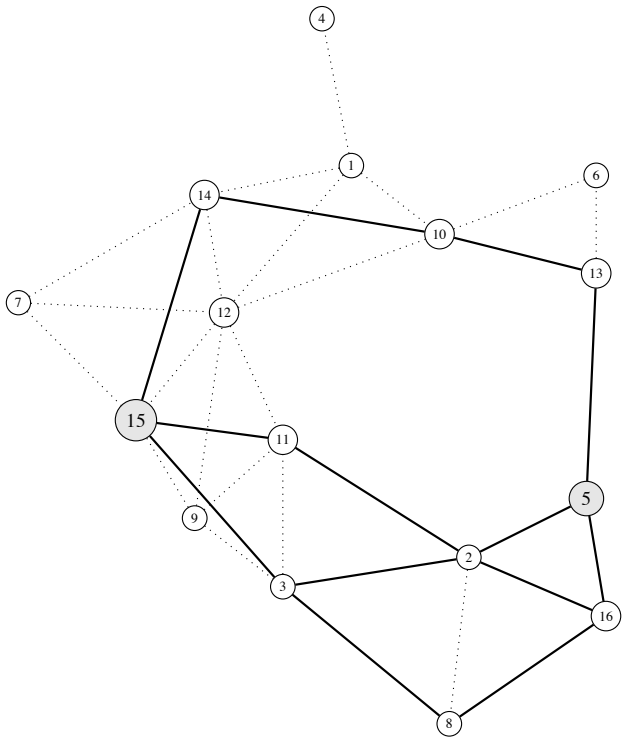

Figura 4.10: Ejemplo de escenario ilustrativo en el que ZD podría tener mejor rendimiento que LD y ND.

Podría por tanto pensarse que se trata de un algoritmo de escasa utilidad pero, bajo las circunstancias adecuadas, puede llegar a lograr mejores resultados en cuanto al rendimiento agregado que las otras soluciones. Una topología que podría causar este efecto se muestra en la Figura 4.10, donde se muestra un grafo de 16 vértices en los que el 15 y el 5 son a fuente y el destino respectivamente. Destacados en negrita aparecen los distintos enlaces que formarían parte de los caminos hallados por los algoritmos multipath, y que se listan a continuación:

a) Para el algoritmo *link-disjoint*:

$15 \rightarrow 11 \rightarrow 2 \rightarrow 5$

$15 \rightarrow 3 \rightarrow 2 \rightarrow 16 \rightarrow 5$

$15 \rightarrow 12 \rightarrow 10 \rightarrow 13 \rightarrow 5$

b) Para el algoritmo *node-disjoint*:

$15 \rightarrow 11 \rightarrow 2 \rightarrow 5$

$15 \rightarrow 3 \rightarrow 8 \rightarrow 16 \rightarrow 5$

$15 \rightarrow 12 \rightarrow 10 \rightarrow 13 \rightarrow 5$

c) Para el algoritmo *zone-disjoint*:

$15 \rightarrow 11 \rightarrow 2 \rightarrow 5$

$$15 \rightarrow 14 \rightarrow 10 \rightarrow 13 \rightarrow 5$$

Viendo la Figura 4.10 se ve claramente que los caminos devueltos por LD y ND están relativamente próximos entre si, con lo que podemos deducir que el nivel de contención y el número de colisiones será elevado, lo que generará un cuello de botella para los paquetes que se transmitan a través de dichas rutas.

Dado que ZD evita el "contacto" entre nodos pertenecientes a distintas rutas, elimina ese efecto "embudo" que se producía con los otros algoritmos, consiguiendo de esta manera un ligero aumento del rendimiento total cuando se emplea MPTCP sobre un escenario con una sola interfaz de comunicación.

5
Network-Coding

5.1 Introducción

En el ámbito de las redes de telecomunicaciones, especialmente en las inalámbricas, la técnica conocida como *network-coding* se ha demostrado que[21] puede beneficiar, en cierta medida, tanto el *throughput* de la red, como su fiabilidad y también movilidad en redes con topologías cambiantes. Esta metodología consiste en aprovechar, siempre que sea posible, la combinación de dos (o más) flujos de datos para codificarlos juntos en un solo paquete, y retransmitirlo hacia los nodos destino donde eventualmente serán decodificados y procesados. Network-coding aprovecha de manera inteligente las diferentes características que presentan los medios inalámbricos frente a los cableados:

a) Cuando uno de los nodos retransmite un paquete de datos, hay una gran probabilidad de que alguno de los vecinos cercanos pueda captar dicha transmisión y convertirse, de esta manera, en un *next-hop* de la ruta a seguir por el paquete para alcanzar su nodo destino.

b) Al tratarse de un "medio compartido", durante las distintas transmisiones de un paquete a lo largo de la red, esta información estará disponible temporalmente para muchos nodos que captarán la señal, a pesar de no ser los destinatarios.

Valiéndose de estas propiedades, network-coding (de ahora en adelante **NC**) puede lograr un aumento considerable del rendimiento, minimizando el número de transmisiones necesarias para transportar diferentes flujos de información en la red, dado que durante parte del tiempo de este proceso, los datos viajarán codificados en un único paquete.

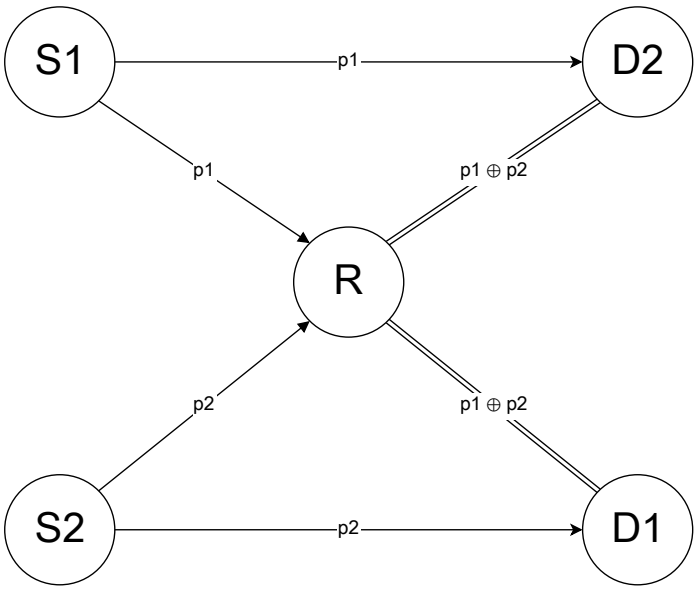

Figura 5.1: Topología en 'X'

5.2 La problemática de network-coding

Una vez presentada la actividad fundamental de NC sobre sistemas inalámbricos, el objetivo que se presenta ahora es doble. Por un lado se encuentra el problema de decidir cuándo es adecuado aplicar esta técnica sobre un grafo en particular y cuándo no. Como se demostrará más adelante, son numerosas las ocasiones en las que resulta innecesario su aplicación, incluso contraproducente, y otras en las que directamente resulta imposible debido a la topología u otras características de la red.

Por otro lado, hay que decidir cuál es el nodo de la red más apropiado para llevar a cabo la codificación de la información de los paquetes entrantes en uno único, que será retransmitido posteriormente hacia los nodos destino. Puede ocurrir también que no sea uno, sino varios los candidatos codificadores en un grafo, por lo que será necesario estudiar esta posibilidad y explotarla en la medida de lo posible. Es necesario destacar una regla de comportamiento que indican Katti et al. en su artículo sobre la arquitectura COPE[20]:

> A la hora de transmitir n paquetes, $p_1, .., p_n$, hacia n nodos destino, $r_1, .., r_n$, un nodo puede codificar (XOR) los n paquetes juntos si y solo si cada destino r_i posee todos los $n-1$ paquetes p_j para $j \neq i$.

Para comprender mejor esta regla se ejemplificará con el grafo en forma de 'X' (Figura 5.1). Este grafo posee dos nodos fuente (S_1, S_2), dos nodos destino (D_1, D_2) y uno intermedio denominado R. Cada una de las fuentes pretende transmitir su paquete de datos p_i a su nodo destino. En circunstancias normales, sin aplicar network-coding y suponiendo que se puede transmitir un solo paquete por cada enlace por unidad de tiempo (u.t.), serían necesarias tres u.t. para completar la restransimisión de ambos paquetes, como se

observa en la Figura 5.2.

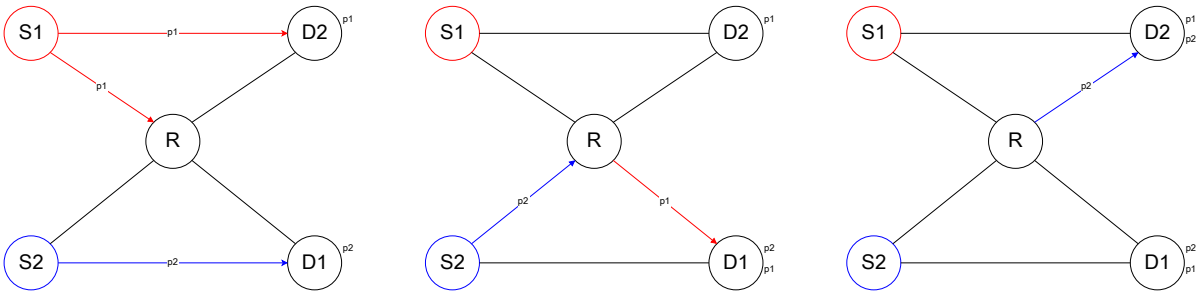

Figura 5.2: Transmisión **sin** network-coding en tres fases

En cambio, aplicando NC en esta red se podría ahorrar una u.t. (Figura 5.3). En el primer instante de tiempo ambos paquetes p_1 y p_2 alcanzarían el nodo destino opuesto, además del nodo intermedio R. El nodo R en este momento puede codificar los dos paquetes en uno sólo. En la segunda u.t. el nodo intermedio transmite el mismo paquete codificado a los nodos destino D_1 y D_2, quienes serán capaces de decodificar la información que esperaban ya que disponen del paquete previo (del instante anterior).

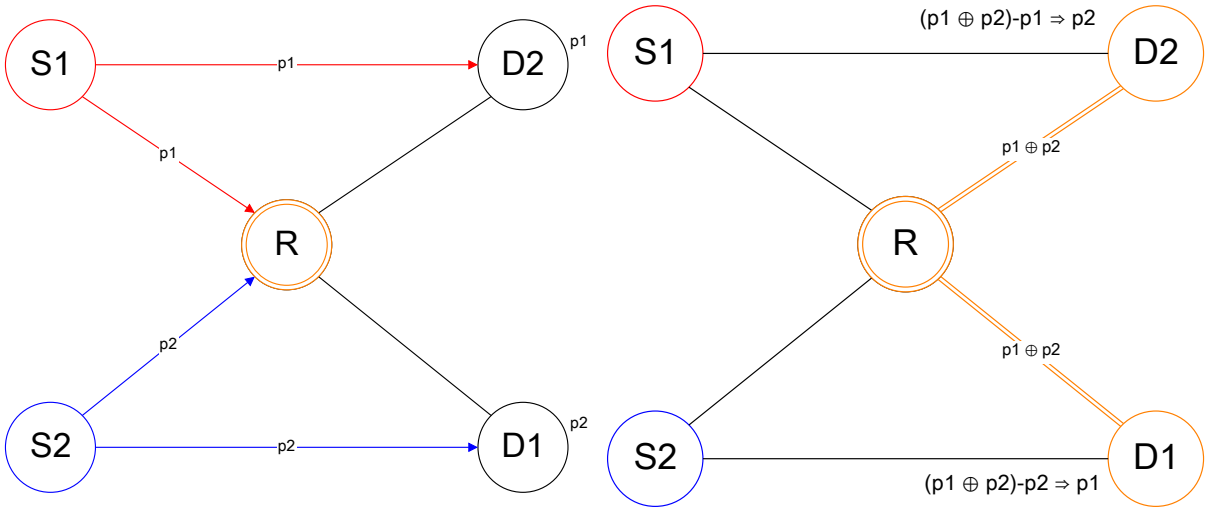

Figura 5.3: Transmisión **con** network-coding en dos fases

5.3 Tipos de rutas en los escenarios con network-coding

Durante todo el trabajo realizado con network-coding siempre se han usado escenarios con dos flujos de información independientes, provenientes de sendos nodos fuente distintos, y dirigidos a otros dos nodos destino, respectivamente. Con esto en mente conviene subrayar un concepto clave del que se hablará mucho a lo largo de los distintos algoritmos implementados: el concepto de ***camino directo*** y ***camino cruzado***. Se define el término "camino directo" como:

> *La secuencia de nodos más corta que conecta uno de los nodos fuente con el nodo destino correspondiente al otro nodo fuente del grafo.*

Mientras que la definición de "camino cruzado" sería la que se ve a continuación:

> *Una secuencia de nodos que conecta uno de los nodos fuente con su correspondiente nodo destino y en la que, además, alguno de los nodos que lo integran coincide con uno (o más) de los nodos que forman el camino cruzado correspondiente al otro nodo fuente.*

Las Figuras 5.4 y 5.5 representan gráficamente un ejemplo de estos dos tipos de rutas dentro de un sistema en el que es posible aplicar network-coding. En la primera pueden observarse los dos tipos de ruta mencionadas, tanto de un nodo fuente como del otro, hacia los nodos destino D_1 y D_2. Después, la conjugación de ambos caminos cruzados en uno de los nodos intermedios del grafo (nodo *1*) y en la última de las imágenes, destacados en color naranja, se encuentran aquellos nodos intermedios considerados candidatos a codificador en esta red.

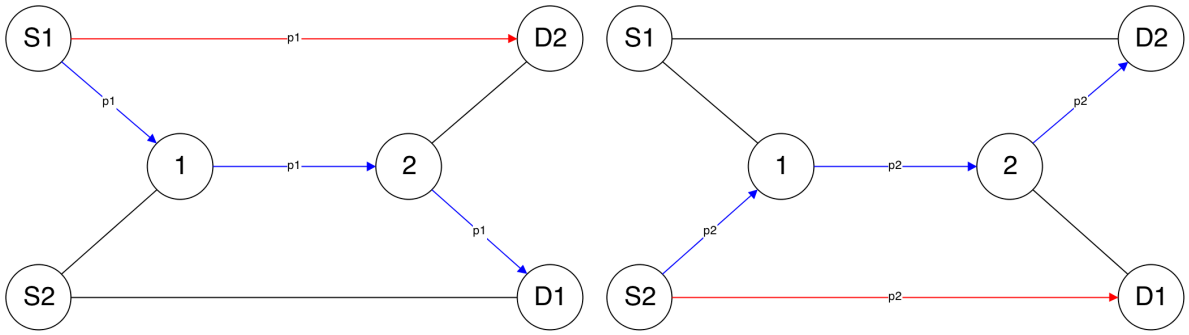

Figura 5.4: Caminos "directo" y "cruzado" en color rojo y azul, respectivamente.

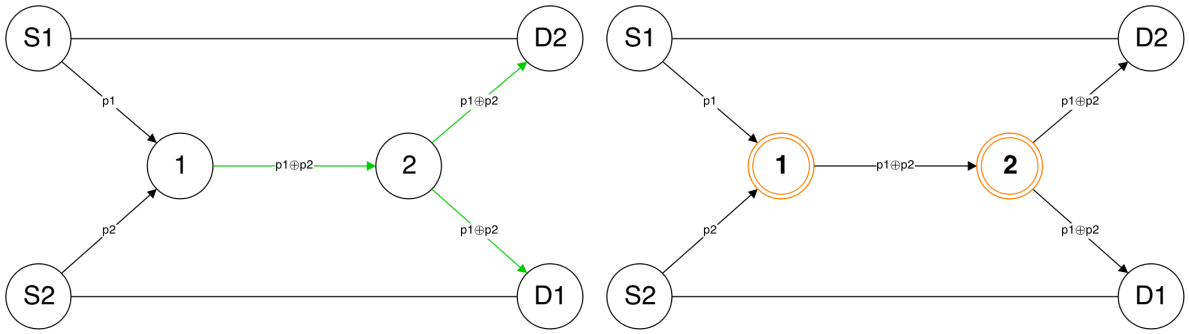

Figura 5.5: Ruta del paquete codificado y posibles nodos codificadores.

Comprender la dualidad de estos caminos y sus propiedades resulta imprescindible para entender el núcleo de los algoritmos desarrollados en este ámbito. A continuación se enumeran las principales propiedades de estas dos clases de rutas:

a) La longitud (en número total de saltos) del "camino directo" ha de ser menor o igual a la del "camino cruzado". En caso contrario se considera innecesaria (*no beneficiosa*) la aplicación de la técnica de network-coding, ya que la transmisión habría terminado para cuando el paquete combinado alcanzase el nodo destino.

b) Por el "camino directo" entre un nodo fuente S_1 y un destino D_2 circula la información necesaria para que dicho destino sea capaz de decodificar la información que viene del nodo S_2.

c) Respecto al "camino cruzado" entre un nodo fuente S_1 y su destino D_1, es necesario que al menos uno de sus nodos coincida con el "camino cruzado" de S_2 con D_2. **El nodo en el cual ambas rutas se cruzan será un candidato a nodo codificador del grafo.** Esta clase de ruta no es única, y será en los últimos algoritmos implementados los que exploten la posibilidad de que existan múltiples "caminos cruzados".

d) Puede haber "caminos cruzados" no válidos, en los que ninguno de sus nodos coincidan con los del otro "camino cruzado". En estas situaciones simplemente se descartarán esos casos y se explorarán más opciones.

e) Cabe la posibilidad de que haya más de un candidato a nodo codificador. En este caso es necesario estudiar uno por uno su *bondad* teniendo en cuenta las posiciones de los nodos fuente para elegir el óptimo.

f) Los nodos fuente y destino de los despliegues con los que se trabajará no podrán asumir el papel de nodo intermedio en ninguna de las rutas estudiadas. Se considera que la función de los nodos fuente es siempre la de enviar información, mientras que la de los nodos destino es la de recibirla y procesarla, pero no retransmitirla.

Un detalle que quizás haya pasado inadvertido es la posibilidad que dos "caminos cruzados" posean longitudes distintas. A priori puede pensarse que, la información transmitida por uno de los nodos fuente en forma de paquete de datos, alcance el nodo codificador antes o después de que lo haga aquel que venga del otro nodo fuente, produciéndose de esta forma cierto *asincronismo* entre transmisiones, y no se pueda llevar a cabo la codificación de los paquetes en el nodo codificador.

Durante el trabajo realizado no se considerarán situaciones como la descrita anteriormente, dado que están ligadas a tecnología de la red. Pero si que se tendrá en cuenta las posiciones de los nodos candidatos a codificador, puesto que una codificación temprana de los datos se traduce en una disminución del tráfico de la red y menor saturación, junto con el resto de beneficios ya comentados de network-coding.

En vista de estos nuevos elementos resulta fácil entender por qué network-coding no es demasiado polivalente. Requiere que el escenario sobre el que se aplique posea una serie de propiedades muy concretas para poder proporcionar beneficio alguno. Por ello, se comenzó el estudio de esta técnica con grafos sencillos como el de "la mariposa"(Figura

5.4) para acabar probando los algoritmos sobre grafos más complejos, con conjuntos de nodos intermedios aleatoriamente situados.

5.4 Algoritmos desarrollados y su evolución

En los siguientes apartados se procederá al análisis en profundidad de las diferentes funciones implementadas así como la progresiva evolución que fueron sufriendo los algoritmos desarrollados. Todos ellos tienen como objetivo fundamental hallar el nodo (o conjunto de nodos) que se encargue de codificar los paquetes de información en un escenario que haga uso de network-coding. La diferencia está en la manera en que lo hacen, en su eficiencia o en el número de candidatos a codificador que son capaces de encontrar.

5.4.1. DFS_NC

Se trata del primer algoritmo desarrollado y también el más sencillo y primitivo en términos de funcionalidad. Se basa principalmente en la utilización del algoritmo de exploración de grafos DFS (*Depth First Search*), que procesa las "ramas del árbol de exploración" en profundidad, como ya se explicó en el capítulo de estado del arte.

Conociendo de antemano los índices de los nodos fuente y destino (S_1, S_2, D_1, D_2), el objetivo de esta función es el de construir los "caminos cruzados" de ambos nodos fuente ($S_1 \rightarrow D_1, S_2 \rightarrow D_2$). Una vez obtenidos estos, el algoritmo los recorre y busca los vértices coincidentes que, como se comentó previamente, serían los candidatos a codificadores de este grafo. El resultado devuelto será una lista formada por este clase de nodos.

El desarrollo de este algoritmo es anterior al estudio sobre los tipos de caminos (directo y cruzado) y su relevancia de cara a una óptima aplicación de network-coding, por lo que aquí no se tienen en cuenta los "caminos directos". El algoritmo tampoco realiza un análisis previo de la viabilidad de network-coding sobre el grafo, a diferencia de los siguientes algoritmos implementados. Esto se traduce en un claro problema de eficiencia al tener que esperar a que termine la función para saber si se puede o no aplicar esta técnica.

5.4.2. BFS_NC

Si el algoritmo anterior se basaba en el uso de DFS para construir las rutas cruzadas de nodos, éste hace lo propio con el BFS (*Breadth First Search)*, algoritmo que como se comentó en apartados anteriores explora el grafo en anchura, dando prioridad a vértices de distintas "ramas" de la misma altura sobre los de niveles inferiores.

La principal diferencia con la función anterior es la capacidad para hallar y construir, en un mismo recorrido del grafo, los "caminos directos y cruzados" de uno de los nodos fuente. Este es un detalle de gran relevancia, pues permite detectar con cierta anticipación si es o no posible aplicar network-coding en el grafo. Un "camino directo" $S_1 \rightarrow D_2$ cuya longitud (en número de saltos) sea mayor que la del "camino cruzado"

$S_1 \rightarrow D_1$ hace innecesario el uso de esta técnica.

El algoritmo recorre el grafo en BFS desde cada uno de los dos nodos fuente, y va construyendo la ruta vértice a vértice, hasta alcanzar un nodo destino. Si éste se trata de un "destino directo" continúa la exploración hasta hallar su correspondiente nodo "destino cruzado". Si, en cambio, encuentra un "destino cruzado" primero, abortaría la ejecución e informaría sobre ello en la salida.

Aunque este algoritmo soluciona el problema de los "caminos directos" seguía teniendo otra limitación, y es la posibilidad de que devuelva **falsos negativos**, es decir, indicar como no-viable el uso de NC en un grafo cuando en realidad sí que es posible su uso. Esto era consecuencia de una insuficiente exploración de la red, ya que los "caminos cruzados" encontrados por este algoritmo eran técnicamente la ruta más corta posible entre un nodo origen S_1 y su destino D_1, partiendo del supuesto de que no se puede atravesar otros nodos fuente o destino del grafo.

En un hipotético caso ideal sería factible hallar el nodo codificador óptimo entre estos dos "caminos cruzados". Pero en términos más realistas es muy probable que no se encuentre dicho nodo y sea necesario analizar otras rutas alternativas, más largas en número de saltos, pero también con una mayor probabilidad de *cruce* entre esos dos caminos.

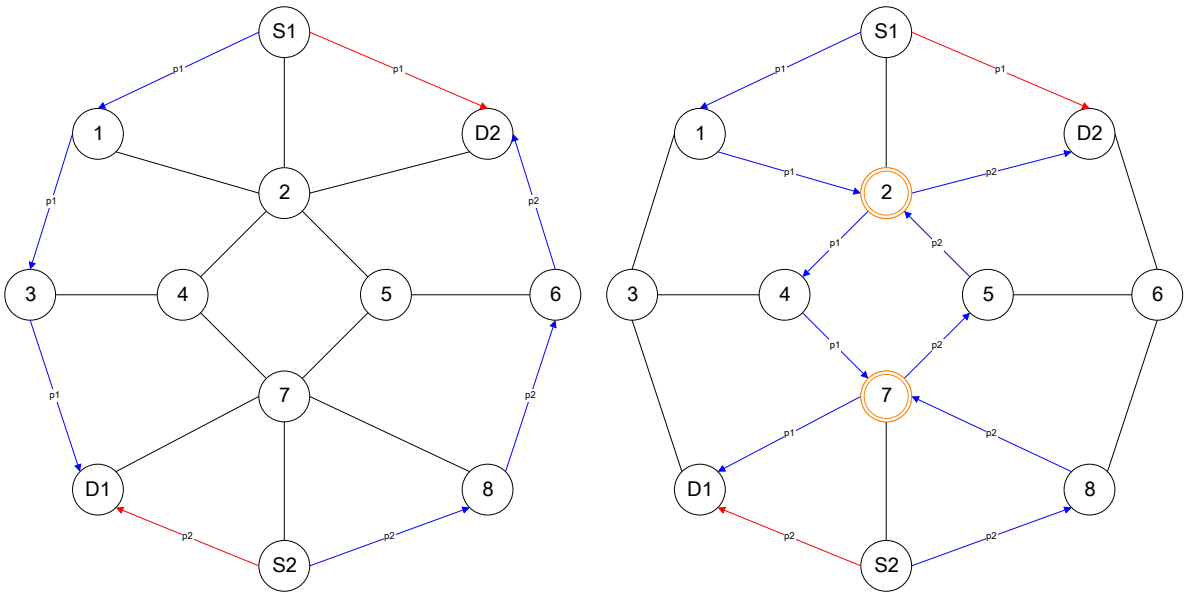

Figura 5.6: Ejemplo de un "falso negativo".

Dicho de otro modo, los "caminos cruzados" de mayor longitud hacen más fácil encontrar más y mejores candidatos a nodo codificador, pero también hacen más ineficiente la transimisión de datos a través de la red puesto que se hace uso de un mayor número de nodos de comunicación intermedios, lo que incrementa el retardo de la misma y la

saturación del sistema. En la Figura 5.6 se observa un ejemplo de este comportamiento, donde los "caminos directos" se muestran en rojo y los "cruzados" en azul. La primera y única opción que nos devolvería el algoritmo *BFS_NC* (en el lado izquierdo de la Figura) sería la de no aplicar NC en este grafo, cuando se ve en el margen derecho una de las muchas opciones existentes que contradicen ese resultado negativo.

5.4.3. BnKSP

El último de los algoritmos implementados es el más completo de todos en términos de funcionalidad, y también el único que logra suplir las pricipales deficiencias de las otras propuestas. Recibe el nombre de *BnKSP* por la conjunción de los algoritmos que internamente lo componen: el ya conocido BFS y el KSP.

KSP o *K-Shortest Paths*[22], es un algoritmo pensado para hallar el conjunto de caminos más cortos entre dos vértices de un grafo dado, siendo K la cantidad de caminos que se desea encontrar. En esencia, consiste en hallar la ruta más corta entre la pareja de vértices especificada y luego, en base a ésta, ir generando otros caminos alternativos entre estos nodos, siempre evitando formar bucles.

En vista de las carencias que presentaba el algoritmo previamente implementado, era necesario buscar una solución que permitiese, de forma sencilla pero eficiente, encontrar "caminos cruzados" alternativos a los que encontraba *BFS_NC* y, de esta manera, abordar el problema de los *falsos negativos* que surgían en la propuesta anterior. El procedimiento que sigue este algoritmo es el siguiente:

1. Mediante el uso del algoritmo BFS determinar en primera instancia, si es viable aplicar network-coding en el grafo, y por otro lado hallar los "caminos directos" del mismo.
2. Mediante el algoritmo KSP, hallar (si es posible) los K-caminos cruzados más cortos y ordenarlos por su longitud.
3. Analizar todos los "caminos cruzados" encontrados en busca de vértices coincidentes y almacenarlos como candidatos a codificador.
4. Comparar los candidatos hallados y seleccionar el mejor, en función de su distancia media (en número de saltos) a los dos nodos fuente.

La razón por la cual se usa BFS y no otro algoritmo de exploración más simple, como por ejemplo el de Dijkstra (implementado como función auxiliar para muchas otras funciones a lo largo de todo el trabajo), es por su capacidad de "barrer" en su anchura todos los vértices del grafo. Este permite, en un recorrido, establecer la ruta más corta al vértice destino opuesto ("camino directo") o al suyo propio, en caso de hallarlo primero. En este último caso la función terminaría, al detectar que la longitud de uno de los "caminos

cruzados" es mayor que la del "camino directo".

Una extensión que pronto se decidió aplicar a este algoritmo fue la de recolectar todos los vértices que se obtuviesen como candidatos a nodo codificador, dado que resultaba interesante conocer posibles alternativas al nodo considerado como óptimo. Los vértices obtenidos previamente de la intersección de los diferentes "caminos cruzados" se almacenarían y ordenarían según su distancia a los nodos fuente del grafo. Esta distancia será calculada en forma de **número de saltos** desde cada nodo fuente hasta el codificador.

Ejemplo práctico

Para ejemplificar todo esto de forma gráfica se utilizará de nuevo el grafo de la Figura 5.6. Sus características lo hacen idóneo para demostrar la funcionalidad del algoritmo *BnKSP*, ya que sus rutas "directas" son obvias y permiten la efectiva aplicación de network-coding sobre esta red. Los nodos fuente (S_1, S_2) y destino (D_1, D_2) están separados, con lo que la intersección de los "caminos cruzados" está asegurada. Otro detalle a destacar es que se trata de un grafo de enlaces bidireccionales, a excepción de los que salen de los nodos fuente y los que llegan a los nodos destino, que son de dirección única. En todo momento se trabajará con *pesos* ideales, con valor 0 si no hay conexión y 1 en caso contrario.

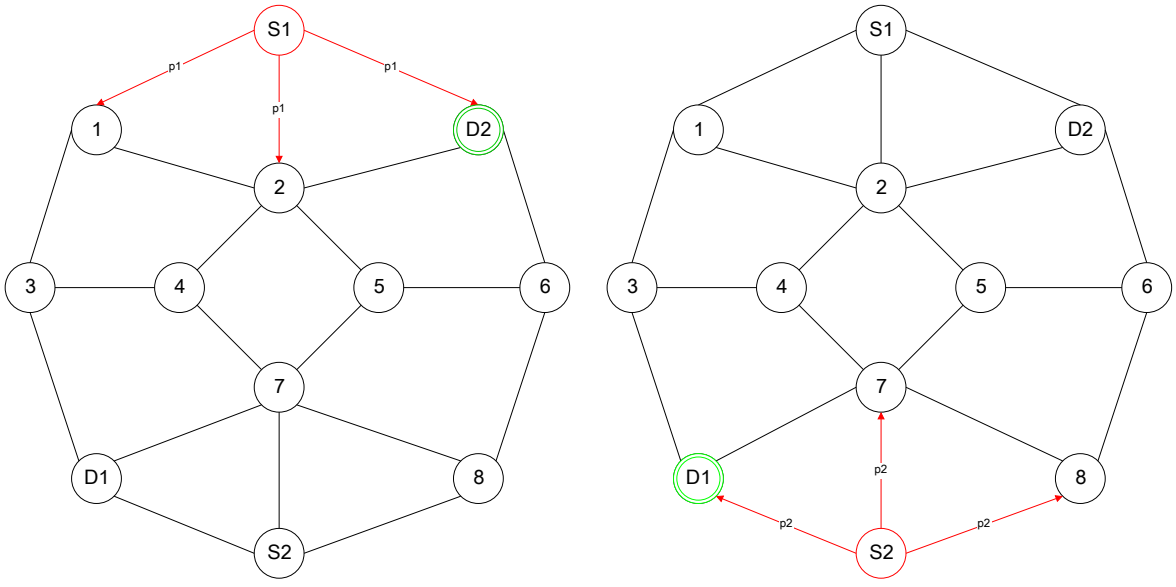

Figura 5.7: Obtención de los "caminos directos".

Comenzando la ejecución del algoritmo con el nodo fuente S_1 como referencia, el primer paso será localizar su correspondiente nodo destino opuesto (5.7). En la organización de vértices del grafo seleccionado ésta es una operación trivial, ya que en la primera iteración del BFS ya encontramos dicho nodo destino a un solo salto de distancia. Por

tanto, el primer "camino directo" del grafo es: $S_1 \rightarrow D_2$. Al haber hallado antes el vértice destino opuesto D_2 que D_1 prosigue la ejecución del algoritmo. El segundo "camino directo" es una operación simétrica a la primera, con $S_2 \rightarrow D_1$ como resultado inmediato.

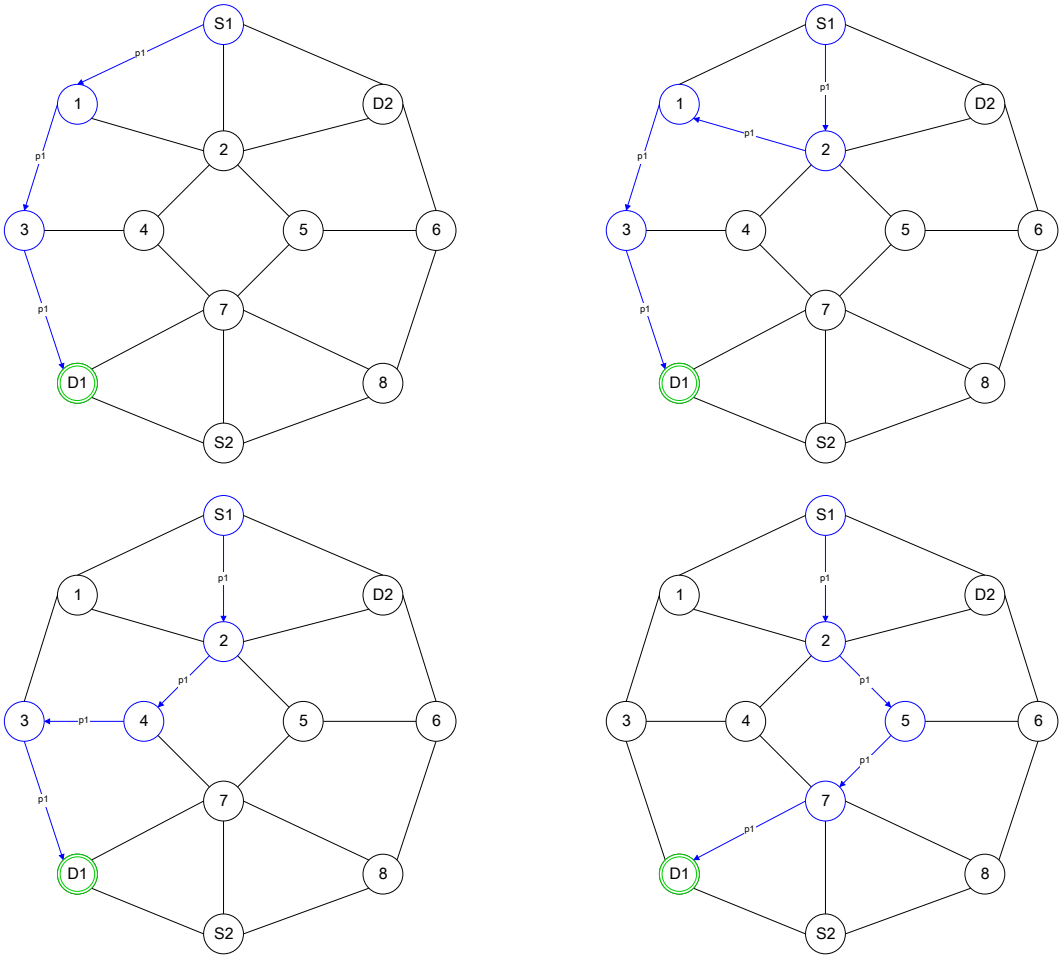

Figura 5.8: Cálculo de los 4 "caminos cruzados" más cortos de S_1.

El siguiente paso es obtener los "caminos cruzados" de ambos nodos fuente, aquellos que conectan con su correspondiente nodo destino. Para mayor simplicidad se tomará un valor de K reducido ($K = 4$). Partiendo del primer nodo fuente y siguiendo el mismo orden del algoritmo KSP obtendríamos las rutas que se muestran en la Figura 5.8. La primera de ellas, que también es la más corta de todas con 3 *saltos* de distancia, es la obtenida directamente mediante Dijkstra. El resto de caminos no son más que derivaciones del primero sin permitir vértices repetidos. De manera homóloga, la Figura 5.9 representa las 4 rutas más cortas procedentes del nodo fuente S_2 con destino D_2.

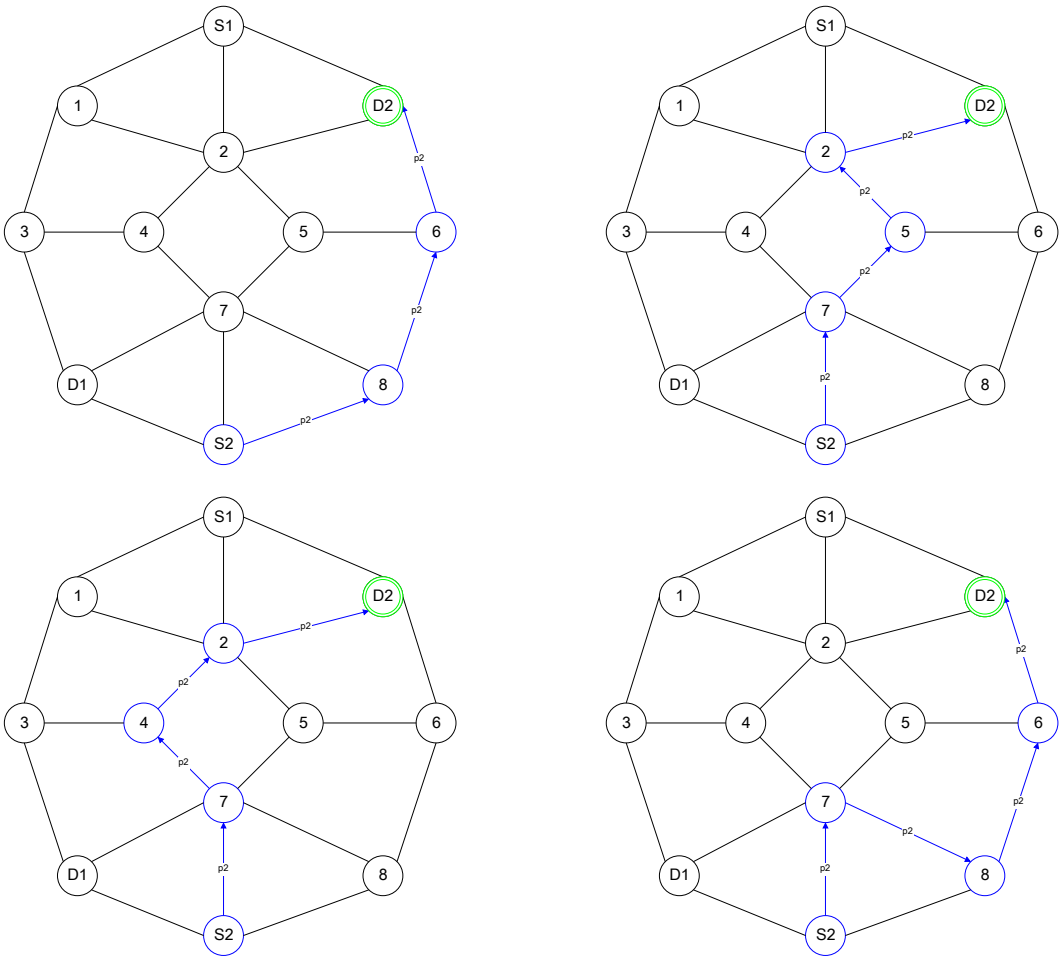

Figura 5.9: Cálculo de los 4 "caminos cruzados" más cortos de S_2.

El último paso en el algoritmo implementado sería el de calcular qué vértices son los más indicados para realizar network-coding en este grafo. Para ello, es necesario recorrer todos los "caminos cruzados" obtenidos en el apartado previo, y localizar aquellos que coincidan en ambas series. En última instancia, solo se tendrá en cuenta la posición más cercana del candidato frente a ambos nodos fuente. La Figura 5.10 muestra solo uno de los muchos casos que pueden darse en la disposición de nodos elegida, con los vértices 2 y 4 como protagonistas. En resumidas cuentas, para un valor de $K = 4$ en este grafo los vértices candidatos a nodo codificador serían los cuatro situados en el centro (2,4,5,7), todos ellos con un *peso* en saltos de 4.

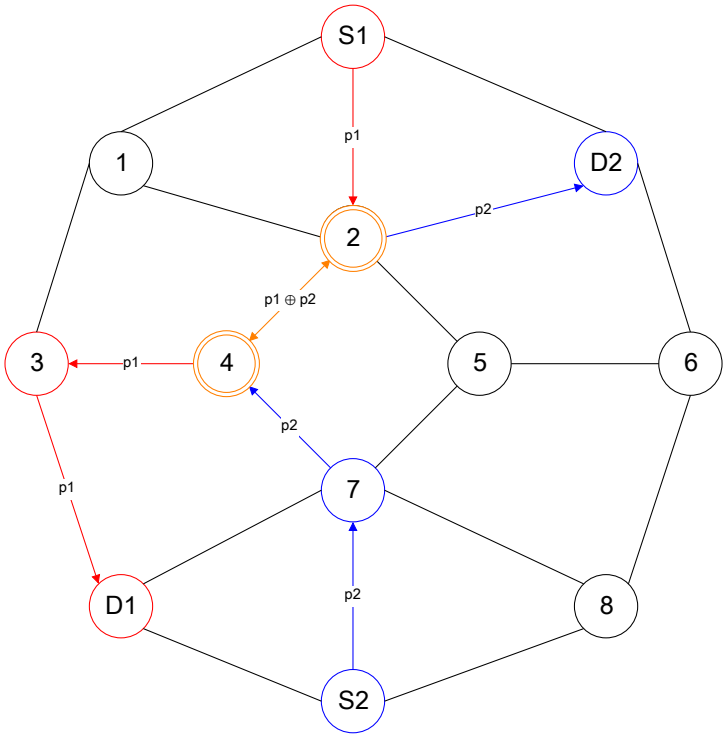

Figura 5.10: Ejemplo de nodos candidatos a codificador.

En una disposición de nodos simétrica como la del ejemplo seleccionado resulta evidente que sean los vértices centrales del grafo los primeros candidatos que se obtengan. Si por ejemplo se aumenta el valor del parámetro K hasta 12, se obtendría como resultado otros candidatos que no se veían antes: 1,3,6 y 8, aunque estos nuevos codificadores estarían situados a una distancia mayor que los calculados anteriormente.

Este ejemplo pone de manifiesto la funcionalidad que se buscaba al usar el algoritmo KSP para ampliar la funcionalidad de network-coding, y es que, al disponer de no sólo una pareja de "caminos cruzados", sino K parejas, se incrementan las posibilidades de aplicar esta técnica sobre un despliegue de nodos adecuado. Además, cuanto mayor es valor del parámetro K, mayor es la garantía de que no se producirán los mencionados "falsos negativos". Al tener un mayor conocimiento sobre la topología de la red y las posibles rutas alternativas entre un nodo fuente y su destino, se es capaz de establecer toda una serie de candidatos a nodo codificador con los que NC pueda ejercer su labor eficientemente.

5.5 Pruebas realizadas

Una vez implementado el algoritmo y visto que su funcionamiento era el esperado en casos simples como el que acaba de verse, se diseñó un banco de pruebas con un esquema similar al usado en el apartado de multipath. El objetivo sería someter el último de los algoritmos desarrollados (*BnKSP*) a escenarios más complejos, en los que los nodos estuvieran ubicados en posiciones aleatorias, y donde los caminos directos y cruzados no

fuesen tan triviales.

La elección de los nodos fuente y destino se haría, igual que en multipath, en base a su distancia a unas coordenadas fijadas de antemano. El área de trabajo será la misma en todas las pruebas, un plano de dimensiones 100x100(u.d.). En este caso existirán siempre dos vertices que harán de nodo fuente y dos que harán de destino. Los primeros situados lo más cerca posible de las coordenadas (20,80) y (20,20) mientras que los segundos cerca de (80,80) y (80,20), buscando, de esta manera, escenarios con una disposición de caminos en 'X', lo que favorecería enormemente las probabilidades de cruce entre caminos.

Las baterías de experimentos estarían compuestas por conjuntos de 1000 grafos, de diferentes cantidades de vértices (16, 32, 64, 128) pero mantiendo constante el radio de cobertura inalámbrica. El parámetro K, número máximo de caminos más cortos a rastrear en los diferentes escenarios, también tiene un papel relevante en las pruebas. Para un análisis en mayor profundidad de algunas variables se trabajó con un escenario con 32 nodos y 20u.d. como medida del radio de cobertura.

Los aspectos más interesantes que se estudiarán en estas pruebas son los siguientes:

1. El procentaje de escenarios en los que es posible aplicar la técnica de network-coding eficientemente.
2. El número total de candidatos a nodo codificador en dichos escenarios.
3. El peso o "bondad" de estos candidatos, medido en función de la distancia total a la que se encuentran de los nodos fuente.
4. Las longitudes de los caminos directos y cruzados de ambos nodos fuente, en términos de número de saltos.

5.5.1. Resultados de los tests

A continuación se presentan los resultados más interesantes de las pruebas efectuadas, representados gráficamente para una mayor facilidad a la hora de interpretar esos datos. La primera de ellas (Figura 5.11) describe el porcentaje de escenarios en los que el algoritmo *BnKSP* devolvió un resultado satisfactorio, en función del del número de nodos del escenario y el valor del parámetro K en ese momento.

Puede apreciarse como en los escenarios con mayor cantidad de nodos es considerablemente más fácil aplicar la técnica de network-coding que aquellos en los que hay menos nodos, estando estos últimos menos afectados por el valor de K. Tal es su influencia que, para el caso de 128 nodos, la probabilidad de poder aplicar NC aumenta en un 35 % cuando se pasa de usar 2 caminos cruzados por nodo fuente a 8.

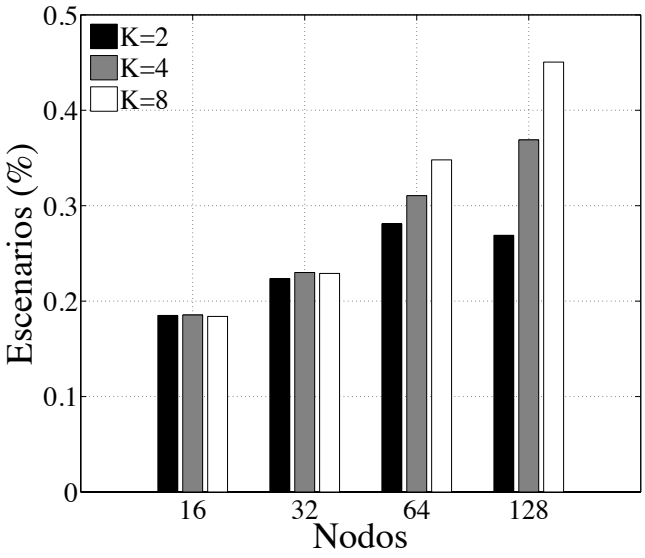

Figura 5.11: Probabilidad de poder aplicar network-coding según el número de nodos.

En vista de estos primeros resultados se deduce que, en aquellos grafos con pocos vértices, una exploración más exhaustiva en busca de más oportunidades de codificación apenas ofrecerá mejores resultados. Por el contrario, los escenarios con una alta densidad de nodos entre fuentes y receptores proporcionan muchos más posibilidades para construir rutas alternativas con las que encontrar nuevas posibilidades de codificación.

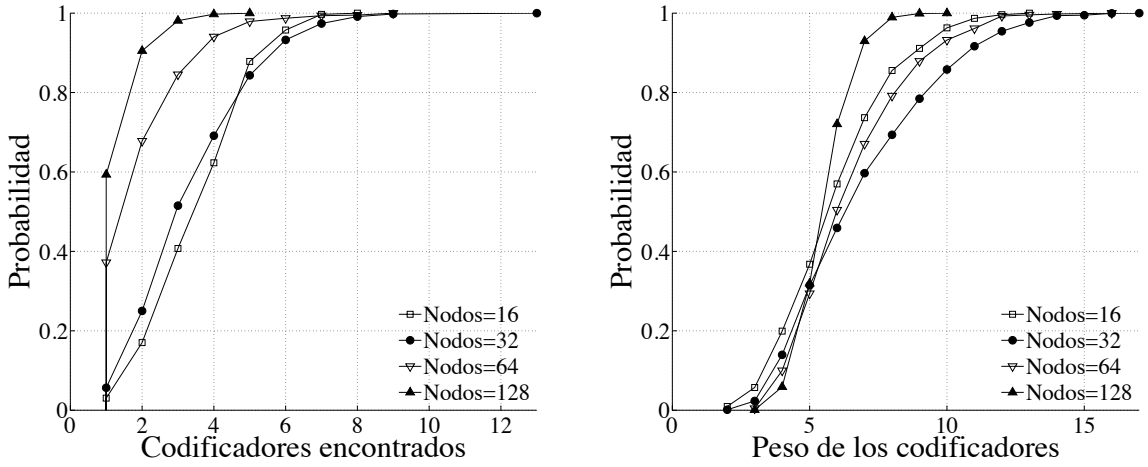

Figura 5.12: Función de probabilidad del número de codificadores hallados y de sus pesos.

La siguiente Figura(5.12), dividida en dos apartados, trata sobre los nodos candidatos a codificador hallados por el algoritmo en diferentes tipos de escenario. Ambas gráficas representan una función de probabilidad acumulada (cdf), siendo el caso de la izquierda la cantidad total de candidatos encontrados y la de la derecha su "peso" o distancia a los nodos fuente. En ambos casos el valor de K fue de 4.

Observando la primera gráfica, la conclusión inmediata que se obtiene es que, a medida que aumenta el número total de nodos en la red, más difícil resulta encontrar nodos codificadores (siempre y cuando el valor de K se mantenga inalterado). Como hipótesis para explicar este comportamiento se podría argumentar que, habiendo una mayor densidad de nodos separando las fuentes y los receptores (sobre todo en los casos de 64 y 128 nodos), las rutas más cortas encontradas entre éstos tienden a ser de poca longitud y más "directas". El algoritmo no busca los caminos cruzados que más tiendan a entrecruzarse, sino aquellos que alcancen lo antes posible el nodo destino. Al haber más nodos, es más sencillo encontrar rutas más directas, lo que dificulta la tarea de encontrar posibles nodos codificadores en los cruces.

Esta hipótesis se ve reforzada al observar (Figura 5.12, derecha) como en los escenarios de 128 nodos, los codificadores hallados tienen una distancia a los nodos fuente ligeramente menor que los del resto de escenarios.

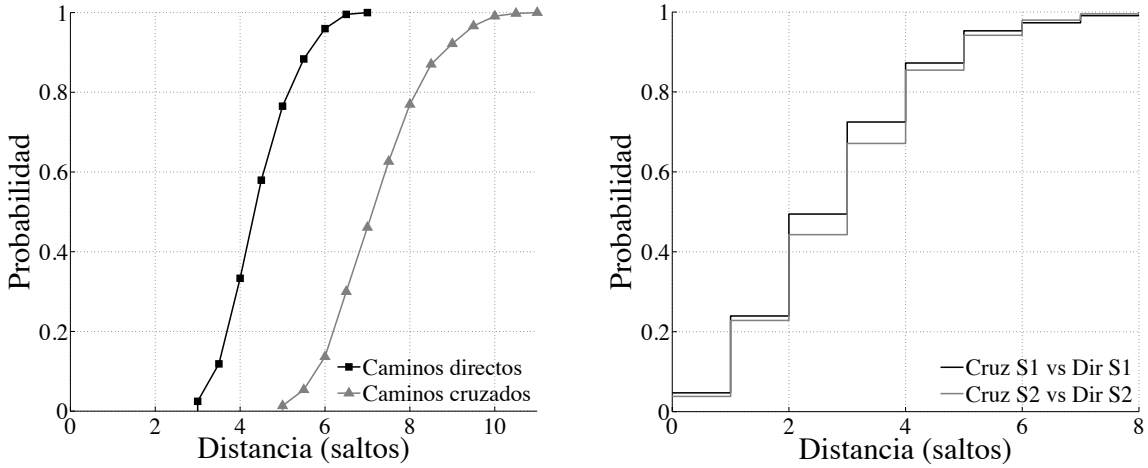

Figura 5.13: Distancias de los caminos directos y cruzados.

En cuanto a las características de los diferentes caminos encontrados por el algoritmo, la primera gráfica de la Figura 5.13 muestra una cdf del número medio de saltos de las rutas directas y cruzadas. Se aprecia claramente como las primeras tienden a ser más cortas que las segundas, aproximadamente dos saltos menos de longitud. La distancia que representa dicha gráfica es la media de ambos caminos directos, en contraste con la de los cruzados preferentes, calculados a partir de un escenario de 32 nodos con una cobertura de 20 (u.d.).

El resultado no sorprende por dos razones distintas. La primera es que es una condición necesaria que las distancias de los caminos directos sean siempre menores a las de los caminos cruzados para poder aplicar network-coding. La segunda razón se entiende al ver la estructura en forma de 'X' de los escenarios generados. Una organización de los nodos realizada adrede para favorecer precisamente este comportamiento de los caminos hallados por el algoritmo.

La gráfica situada a la derecha en la Figura 5.13 refuerza el argumento anterior. La imagen representa, no sólo la citada diferencia de longitudes entre caminos de distinta clase, sino también la semejanza existente en la distancia de las rutas procedentes de los nodos fuente. La diferencia de longitud entre rutas del mismo tipo pero diferente nodo fuente es mínima, como era de esperar, mientras que la de rutas de tipos opuestos puede alcanzar hasta los 8 saltos de diferencia.

6

Conclusiones y líneas de investigación futuras

El trabajo realizado en el marco de este proyecto se ha centrado en dos ramas claramente diferenciadas. Por un lado, se han estudiado e implementado una serie de algoritmos para obtener varios tipos de caminos disjuntos en despliegues inalámbricos multi-salto. Posteriormente, fueron evaluados en base a un conjunto de bancos de pruebas con múltiples escenarios de diversas características.

A tenor de los resultados obtenidos en las pruebas, los algoritmos LD y ND presentan los mejores resultados a la hora de encontrar múltiples rutas disjuntas, explotando las posibilidades que ofrece el encaminamiento multipath. Por otro lado, existe la posibilidad de que dichos algoritmos presenten problemas de interferencias y colisiones entre distintas transmisiones en escenarios de redes malladas reales, especialmente en despliegues con un único canal inalámbrico. El tercero de los algoritmos implementados, ZD, supera parcialmente esta limitación, aunque su capacidad para encontrar rutas disjuntas en un despliegue genérico es considerablemente menor que la de las soluciones mencionadas anteriormente.

De forma paralela, se ha llevado a cabo un estudio en cooperación con otro estudiante del Grupo de Ingeniería Telemática de la UC, comparando el rendimiento del protocolo MPTCP frente al del TCP tradicional. Tanto los algoritmos multipath implementados, como los escenarios aleatorios generados durante las pruebas, tuvieron una participación de relevancia dentro de una línea de investigación que se mantiene activa. Los resultados de este estudio dieron lugar a la creación de dos artículos de investigación enviados al JITEL[17] y la conferencia MONAMI[18], que tendrán lugar entre Septiembre y Octubre de este mismo año.

Por otro lado, se han desarrollado de forma empírica una serie de algoritmos para la aplicación de network-coding en redes inalámbricas multi-salto. El más avanzado de estos algoritmos fue puesto a prueba mediante una serie de experimentos con despliegues aleatorios. El análisis de los resultados obtenidos en las pruebas revela un aumento en el número de escenarios en los que es posible usar esta técnica cuando se aplica el algoritmo implementado, sobre todo en aquellos con mayor número de nodos. Se observa una relación directa entre el número de escenarios en los que es factible aplicar NC y el hecho de realizar una exploración más exahustiva de la red, en busca de una mayor cantidad de oportunidades de codificación.

Como posibles futuras líneas de investigación en el apartado de multipath, sería interesante explorar más el potencial del algritmo ZD para hallar rutas "verdaderamente" independientes, y buscar otros esquemas de encaminamiento, también capaces de encontrar caminos disjuntos, pero asimismo teniendo en cuenta las características intrínsecas de las transmisiones inalámbricas. Otros aspectos a valorar podrían ser el hecho de que algunos nodos de los escenarios no permaneciesen estáticos, o que la calidad de sus enlaces estuvieran sujetos a fluctuaciones, pudiendo dificultar o incluso impedir la transmisión de información. En definitiva, buscar otro tipo de soluciones capaces de adaptarse a topologías cambiantes.

En cuanto al bloque de network-coding, sería interesante buscar una solución más eficiente basada en los mismos principios que el último de los algoritmos desarrollados (*BnKSP*). Una búsqueda de caminos directos y cruzados simultánea y programación paralela, haciendo uso de los procesadores *multicore* y su capacidad para trabajar con diferentes hilos de ejecución a la vez, son sólo dos propuestas que mejorarían el rendimiento del algoritmo. Otro aspecto a explorar en un futuro sería el de partir de un despliegue con más de dos nodos fuente y dos destino, aunque en este caso, la dificultad a la hora de encontrar posibles nodos codificadores o de estudiar la "rentabilidad" de NC bajo tales circunstancias, aumentaría la complejidad del algoritmo exponencialmente.

Existe un aspecto importante a destacar en la solución de network-coding implementada en este trabajo, aunque perteneciente a un ámbito más relacionado con los protocolos de enrutamiento, que con la forma en la que se determina la posibilidad de aplicar NC en un despliegue dado de forma algorítmica.

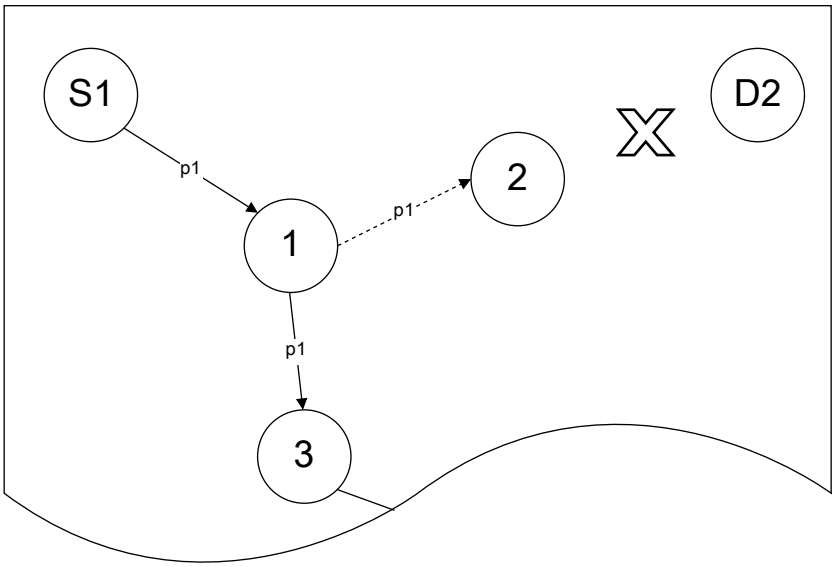

Figura 6.1: El problema del enrutamiento de los caminos directos.

El problema en cuestión tiene que ver con la forma en la que los diferentes nodos de la red encaminan los paquetes provenientes de un nodo fuente u otro. Cuando un nodo

fuente S_1 desea transmitir un paquete p_1 a un nodo destino D_1 los demás terminales de la red establecen, a partir de su tabla de rutas y de las cabeceras de p_1, el siguiente nodo en su ruta hacia su destino. Ésta es la forma habitual de retransmitir paquetes que tienen las redes multi-salto.

El problema surge cuando el citado paquete p_1 tiene como nodo destino D_1, pero a la vez se pretende que alcance previamente el nodo D_2, con el fin de poder aplicar NC. A priori, ningún nodo de la red transmitirá ese paquete con dirección a D_2 de forma explícita (siguiendo de esta manera el "camino directo" de S_1), dado que su destino es, realmente, D_1. La Figura 6.1 refleja esta situación, donde el nodo **2** "oye" el paquete p_1, pero no lo retransmite a D_2 porque su tabla de rutas no está configurada para que lo haga.

Se trataría pues, de otro aspecto a tener en cuenta con vistas a trabajar en un futuro con una solución basada en NC.

7
Bibliografía

[1] T. Braun et al. "Traffic and QoS Management in Wireless Multimedia Networks," *Lecture Notes in Electrical Engineering* Volume 31, 2009.

[2] S. Waharte, R. Boutaba "Totally Disjoint Multipath Routing in Multihop Wireless Networks," School of Computer Science, University of Waterloo, Canada.

[3] N. Meghanathan "Stability and Hop Count of Node-Disjoint and Link-Disjoint Multi-Path Routes in Ad Hoc Networks," in *Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on, 2007.*

[4] N. Meghanathan "Graph Theory Algorithms for Mobile Ad Hoc Networks," *Informatica 36:185–199*, Vol.36, No.4, December 2012.

[5] N. Meghanathan "Performance Comparison Of Link, Node And Zone Disjoint Multi-Path Routing Strategies And Minimum Hop Single Path Routing For Mobile Ad Hoc Networks," *International Journal of Wireless and Mobile Networks* Vol.2, No.4, Jackson State University, USA, November 2010.

[6] R. Ahlswede et al. "Network Information Flow," *IEEE Transactions on Information Theory*, Vol.46, No.4, July 2000.

[7] S.-Y. R. Li, R. W. Yeung and N. Cai "Linear Network Coding," *IEEE Transactions on Information Theory*, Vol.49, No.2, February 2003.

[8] D.S. Lun et al. "Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding," *IEEE INFOCOM*, 2005.

[9] Sachin Katti et al. "XORs in The Air: Practical Wireless Network Coding," in *SIGCOMM'06*, Pisa, Italy, September 2006.

[10] P. Erdös and A. Rényi "On Random Graphs I" in Publ. Math. Debrecen 6 (1959).

[11] E.N. Gilbert "Random Graphs" Ann. Math. Statist. Volume 30, Number 4 (1959).

[12] E.W. Dijkstra "A note on two problems in connexion with graphs" *Numerische Mathematik*, Volume 1, Issue 1, 1959.

[13] Jin Y. Yen "Finding the K Shortest Loopless Paths in a Network" *Management Science*, Vol.17, No.11, Theory Series (Jul., 1971), pp. 712-716.

[14] N. Meghanathan "Graph Theory Algorithms for Mobile Ad Hoc Networks," *Informatica 36:185–199*, Vol.36, No.4, December 2012.

[15] N. Meghanathan "Performance Comparison Of Link, Node And Zone Disjoint Multi-Path Routing Strategies And Minimum Hop Single Path Routing For Mobile Ad Hoc Networks", *International Journal of Wireless and Mobile Networks* Vol.2, No.4, Jackson State University, USA, November 2010.

[16] Network Simulator 3 `http://www.nsnam.org/overview/what-is-ns-3`

[17] C. Rabadán, P. Garrido, D. Gómez, R. Agüero. "Algoritmos y técnicas multi-camino para la mejora del rendimiento de TCP sobre redes malladas inalámbricas". Enviado a las *XI Jornadas de Ingeniería Telemática, JITEL'13*, a celebrar en Granada, entre el 28 y 30 de octubre de 2013.

[18] D. Gómez, C. Rabadán, P. Garrido, R. Agüero. "Multipath Algorithms and Strategies to Improve TCP Performance over Wireless Mesh Networks". Enviado a la *5th International Conference on Mobile Networks and Management, MONAMI'13*, a celebrar en Cork (Irlanda), entre el 23 y 25 de septiembre de 2013.

[19] Sachin Katti et al. "XORs in The Air: Practical Wireless Network Coding," in *SIGCOMM'06*, Pisa, Italy, September 2006.

[20] Sachin Katti et al. "The Importance of Being Opportunistic: Practical Network Coding for Wireless Environments," *Allerton Annual Conference on Communication, Control and Computing*, 2005.

[21] Marek Konieczny "Network coding in wireless environment," in *Seminar on Internetworking*, Helsinki University of Technology, 2008.

[22] Jin Y. Yen "Finding the K Shortest Loopless Paths in a Network" *Management Science*, Vol.17, No.11, Theory Series (Jul., 1971), pp. 712-716.