**ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN**

**UNIVERSIDAD DE CANTABRIA**

*Trabajo Fin de Grado*

# ALGORITMOS PARA LA BÚSQUEDA DE MÚLTIPLES RUTAS SOBRE REDES INALÁMBRICAS MALLADAS MULTI-INTERFAZ

## (Multipath algorithms over wireless mesh multi-interface networks)

**Para acceder al Título de**

***Graduado en Ingeniería de Tecnologías de Telecomunicación***

**Autor: Jaime Piris Ruiz**
**Director: Ramón Agüero Calvo**

**Julio 2015**

# Agradecimientos

*Agradecer a mis tutores, Ramón y Pablo, haberme prestado tanta ayuda y haber respondido todas mis dudas durante la realización de este trabajo, de forma que haberlo realizado a distancia no ha supuesto ningún problema.*

*A mis padres, por haberme apoyado en las decisiones que he tomado aunque haya sido difícil, y haber estado siempre encima de mí para que no dejase el trabajo de lado.*

*Y a mis amigos, que siempre me animan cuando un día las cosas no han ido bien y son especialistas en distraerme cuando tendría que estar trabajando.*

# Índice de la memoria

Índice de figuras……...6

Índice de fórmulas……7

Resumen Ejecutivo……8

Executive Summary……9

**Capítulo 1: Introducción……10**

Objetivos del trabajo……11

Bloque de transformación de la red……...11

Bloque de implementación de los algoritmos multicamino……11

Bloque de simulación……...12

Descripción del flujo de trabajo……...12

**Capítulo 2: Redes malladas inalámbricas de múltiples canales……14**

Introducción a las redes multisalto……15

Protocolos de encaminamiento multisalto……15

Redes malladas multicanal: MIC y LIBRA……17

Supuestos iniciales……..18

Tipos de protocolos de enrutamiento……....18

Requisitos de los protocolos de enrutamiento……18

Funciones Path Weight……...19

Conceptos de inter-flow e intra-flow……..20

IRU……..21

CSC……...22

MIC……23

LIBRA……24

Implementación del protocolo……28

Conclusiones……..30

**Capítulo 3: Algoritmos Multicamino. El protocolo MPTCP……..31**

Introducción……...31

El protocolo MPTCP……32

Consideraciones de diseño……32

Objetivos……...32

MPTCP en la pila de protocolos……33

Funcionalidades…….....33
Algoritmos Multicamino……34
Algoritmo de Dijkstra.……..……35
Link Disjoint……..36
Node Disjoint……37
Zone Disjoint……..38
Implementación de los algoritmos……..…39
**Capítulo 4: Simulación de escenarios mediante el simulador NS-3. Análisis de resultados…….....……..…...40**
Consideraciones previas…….……..……..41
Disposición de los nodos en el espacio……43
Selección de los canales…….45
Simulaciones realizadas…….45
Redes con canales uniformes……45
Link y Node Disjoint……45
Zone Disjoint……49
Valoración general de los protocolos……...50
Redes con canales no uniformes……..52
Rendimiento……54
**Capítulo 5: Conclusiones y líneas de investigación futuras……56**
**Capítulo 6: Bibliografía y referencias……57**

# Índice de figuras

**Capítulo 1:**

1.1 Descripción del flujo de trabajo……13

**Capítulo 2:**

2.1 Concepto de isotonicidad……...19
2.2 Inter-flow……...20
2.3 Intra-flow……...21
2.4 Isotonicidad en MIC……24
2.5 Aproximación a la virtualización……25
2.6 Nodo virtualizado completamente……26
2.7 Red completa transformada……27
2.8 Esquema de la implementación del proceso……...29

**Capítulo 3:**

3.1 MPTCP en la pila de protocolos……33
3.2 Algoritmo de Dijkstra……..35
3.3 Ejecución de Link Disjoint……...36
3.4 Ejecución de Node Disjoint……37
3.5 Ejecución de Zone Disjoint……38
3.6 Esquema de funcionamiento de los algoritmos……..39

**Capítulo 4:**

4.1 Adecuación del formato de las rutas al simulador NS……..42
4.2 Distribución aleatoria de nodos……43
4.3 Distribución que garantiza la existencia de una ruta……44
4.4 LD sobre red de canales uniformes……46
4.5 ND sobre red de canales uniformes……46
4.6 Caso específico de LD……47
4.7 Rendimiento de TCP sobre canales uniformes……..48
4.8 Rendimiento de LD sobre canales uniformes……48

4.9 ZD sobre red de 3 canales……49
4.10 Rendimiento sobre canales uniformes……50
4.11 Ganancia relativa sobre canales uniformes……51
4.12 LD sobre canales aleatorios……52
4.13 Número medio de rutas encontradas en canales aleatorios……53
4.14 Rendimiento de LD sobre canales aleatorios……54
4.15 Rendimiento de ZD sobre canales aleatorios……54
4.16 Ganancia relativa sobre canales aleatorios……55

# Índice de fórmulas

Capítulo 2:

2.1 Definición de isotonicidad……19
2.2 Métrica IRU……21
2.3 Distancia euclídea entre nodos……22
2.4 Métrica CSC con un predecesor……22
2.5 Métrica CSC con dos predecesores……22
2.6 Métrica MIC……23
2.7 Nodos virtuales por nodo real……27
2.8 Total de nodos virtuales……27

Capítulo 4:

4.1 Canales LD y ND en redes homogéneas……45
4.2 Ganancia Relativa……51

# Resumen ejecutivo

En los últimos años, la comunidad científica se ha interesado cada vez más por las posibilidades que ofrecen las redes malladas. Un caso particular de estas redes son aquellas cuyos nodos poseen múltiples interfaces. Hasta ahora, cualquier terminal que quería comunicarse con otro tenía que establecer una conexión a nivel de transporte (en la gran mayoría de los casos, TCP) a través de la cual transmitir.

Sin embargo, recientemente el grupo IETF ha comenzado a trabajar sobre una nueva forma de comunicación: *Multipath TCP*. Debido a que los terminales de comunicación de hoy en día incorporan cada vez más interfaces con las que acceder al medio, parece que seguir utilizando una sola conexión no es lo más óptimo. Mediante este nuevo protocolo, se pretende que las aplicaciones sean capaz de repartir su tráfico entre varias conexiones de nivel de red, lo cual debería -en teoría- mejorar la calidad de las comunicaciones tanto en velocidad como en seguridad y robustez.

En este trabajo de fin de grado implementaremos una serie de algoritmos conocidos como algoritmos *Disjoint*, que nos permitirán repartir el tráfico generado por un nodo y hacerlo llegar por varios caminos a su destino. Para ello, previamente nos apoyaremos en un protocolo de balanceo de carga conocido como *Load and Interference Balanced Routing Algorithm*, con el cual reflejaremos el comportamiento de las redes multicanal inalámbricas.

Por último, realizaremos un conjunto de simulaciones mediante la popular herramienta Network Simulator 3, para recoger pruebas estadísticas que apoyen el empleo de las tecnologías multi-camino en las redes malladas multi-interfaz.

**Palabras clave**

Redes inalámbricas, redes malladas, multi-interfaz, balanceo de carga, LIBRA, algoritmos multi-camino, MPTCP, simulador NS-3.

# Executive summary

In recent years, the scientific community has shown interest for the possibilities that mesh networks can provide. A special case of these networks are those whose nodes possess multiple interfaces. Until now, any terminal that wanted to communicate with another terminal had to establish a Transport layer connection (in most cases, TCP), through which it can transmit.

However, the IETF group has recently started to work on a new form of communication: Multipath TCP. Due to communication terminals nowadays include an increasing number of interfaces from which they are able to access the media, it seems that using only one connection is not the best method. Through this new protocol, it is intended that application will be able to distribute their traffic between various Network layer connections, which should -theoretically- improve the quality of the communications both in speed and in security and robustness.

In this final project we will implement a series of algorithms known as the Disjoint algorithms, which will allow us to distribute the traffic generated by a node and deliver it to its destination through multiple paths. In order to achieve this, we will have previously used a balancing protocol known as *Load and Interference Balanced Routing Algorithm*, thanks to which we will reflect the behaviour of the wireless multichannel networks.

Finally, we will run a series of simulations with the popular tool Network Simulator 3; to gather statistical evidence that supports the use of multi-path technology in multi-interface mesh networks.

**Keywords**

Wireless networks, mesh networks, multi-interface, load balancing, LIBRA, multipath algorithms, MPTCP, Network Simulator 3.

# Capítulo 1: Introducción

El trabajo que se presenta está dividido en tres bloques diferenciados, de los cuales nos serviremos para comprobar la eficiencia del protocolo *Multi Path TCP* sobre redes inalámbricas multicanal.

El primero de dichos bloques consistirá en desarrollar un conjunto de funciones que nos permitan aproximar el funcionamiento de una red inalámbrica multicanal. Para ello, realizaremos un proceso conocido como virtualización de los nodos, de forma que obtendremos una representación fiel del comportamiento de este tipo de redes en cuanto a las dificultades que presenta un entorno inalámbrico. Nos apoyaremos en un conjunto de métricas que modelarán las interferencias entre los distintos nodos.

A continuación seguiremos con la implementación sobre la red transformada de tres algoritmos de búsqueda de múltiples caminos, como son el Link-Disjoint, el Node-Disjoint y el Zone-Disjoint. Todos ellos están basados en el algoritmo de Dijkstra y tienen como objetivo encontrar rutas independientes dentro de una red, variando en cada caso la restricción a la hora de establecer dichas rutas.

Estas dos primeras partes se programaron empleando el lenguaje C++. El motivo de la elección de este lenguaje es su gran capacidad para optimizar recursos al ser capaz de acceder a la memoria a bajo nivel, la posibilidad de crear módulos independientes entre sí mediante la creación de clases y la familiaridad previa con el lenguaje C.

Por último, emplearemos el simulador NS-3 junto con las funciones previamente elaboradas para comprobar el rendimiento que obtenemos al aplicar los algoritmos multicamino. Esta herramienta nos permitirá realizar un gran número de simulaciones con el fin de analizar la viabilidad de los escenarios propuestos a partir de las estadísticas de rendimiento que nos proporciona.

## Objetivos del trabajo

1. Aproximar el comportamiento de las redes de múltiples canales mediante la transformación de nodos reales en virtuales y el empleo de métricas que tengan en cuenta las restricciones intrínsecas de medios inalámbricos, empleando como entrada tanto topologías definidas por usuario como aleatorias y con diferentes características de radio de cobertura y número de interfaces por nodo.

2. Comprobar la viabilidad y rentabilidad de emplear el protocolo *Multi-path Transfer Control Protocol* en redes inalámbricas malladas multisalto (como es el caso de las Wireless Sensor Networks) en oposición a los protocolos de transporte empleados actualmente.

3. Elaborar un conjunto de algoritmos y funciones que permitan aplicar algoritmos multicamino a dichas redes.

4. Realizar un conjunto relevante de simulaciones y representar sus resultados gráficamente para elaborar conclusiones con fundamento acerca de las ventajas y desventajas del uso del protocolo MPTCP.

A continuación expondremos más en detalle los objetivos de cada bloque:

### Bloque de transformación de la red

En este primer bloque partiremos de una red, bien especificada por el usuario o bien generada aleatoriamente, en la que un conjunto de nodos repartidos en el espacio incorporan una serie de canales que comparten con uno o más vecinos. El objetivo será transformar esa red básica en una red "virtual" que simule el comportamiento de esas múltiples interfaces.

Para ello, cada nodo del grafo original será dividido en un conjunto de nodos a los que llamaremos nodos virtuales, y que serán de varios tipos, según la función que realicen dentro del grafo: recepción, envío, generación o consumo de paquetes. De este modo podremos simular el comportamiento real de un nodo multi-interfaz.

Una vez determinados los nuevos nodos virtuales creados, utilizaremos un conjunto de métricas para determinar los pesos de los enlaces entre ellos. Tendremos en cuenta a la hora de utilizar estas métricas varias variables: el radio de cobertura de los nodos, los canales empleados por otros nodos, el canal que emplea el propio nodo…

### Bloque de implementación de los algoritmos multicamino

El objetivo de esta parte del trabajo será implementar en C++ los tres algoritmos de enrutamiento que queremos emplear para descubrir caminos MPTCP, a saber:

- Link-Disjoint
- Node-Disjoint
- Zone-Disjoint

Los tres algoritmos tienen el mismo propósito, y además todos se apoyan sobre uno de los algoritmos de búsqueda de camino de coste mínimo más extendidos: el algoritmo de Dijkstra. Por tanto, una de las fases esenciales será la implementación de dicho algoritmo. Sin embargo, los algoritmos disjoint se diferencian en el número de restricciones que imponen a la hora de elegir las rutas: hablaremos más de ello en el capítulo correspondiente a este bloque.

**Bloque de simulación**

En la última parte del trabajo se relizará un conjunto de simulaciones que reflejen el rendimiento del protocolo MPTCP sobre redes malladas inalámbricas multi-interfaz. Para ello se generarán escenarios de una forma pseudoaleatoria y se introducirán tanto su topología como sus tablas de rutas en el simulador NS-3. El objetivo de este bloque, por tanto, será obtener resultados de su rendimiento, comprobando la validez de todas las teorías expuestas en los dos capítulos anteriores.

**Descripción del flujo de trabajo**

A la hora de plantear el trabajo, y tras elegir C++ como lenguaje de programación, nos encontramos con el primer punto importante del mismo: describir una secuencia de acciones que nos permitan alcanzar nuestro objetivo de aplicar los algoritmos multicamino a redes malladas inalámbricas con múltiples canales. Aprovechando las librerías y clases de C++ y su interacción con archivos elaboramos la siguiente solución:

a. En primer lugar, crear un sistema de entrada con el que recoger datos de un grafo sobre el que trabajar. Este grafo puede ser generado aleatoriamente dentro de unos límites espaciales establecidos, o bien introducido directamente por el usuario mediante un fichero de entrada. Los atributos de cada nodo incluyen su posición en el espacio bidimensional y el uso de los canales inalámbricos (hasta cuatro en un principio).

b. Como segundo paso, implementar un módulo que realice la operación de transformación del grafo. El acceso al sistema de ficheros nos permite generar a lo largo de este proceso un conjunto de salidas que pueden resultar útiles tanto para el desarrollo del trabajo como para el análisis de resultados, como son los correspondientes a aplicar las métricas LIBRA (de las cuales hablaremos en el próximo capítulo). Asimismo, se generará un fichero en el que almacenar el grafo transformado completo.

c. El tercer bloque se encargará de aplicar los algoritmos disjoint al grafo generado en el punto anterior, además de construir unos ficheros de salida del formato adecuado, que se pasarán al simulador NS-3, el cual será para nosotros una caja negra, y no será necesario comprender su funcionamiento para aprovecharnos de sus características.

d. Por último recogeremos los resultados del simulador y los analizaremos estadísticamente, apoyándonos en un conjunto de gráficos generados con Matlab.

Este proceso se representa gráficamente en el siguiente diagrama:

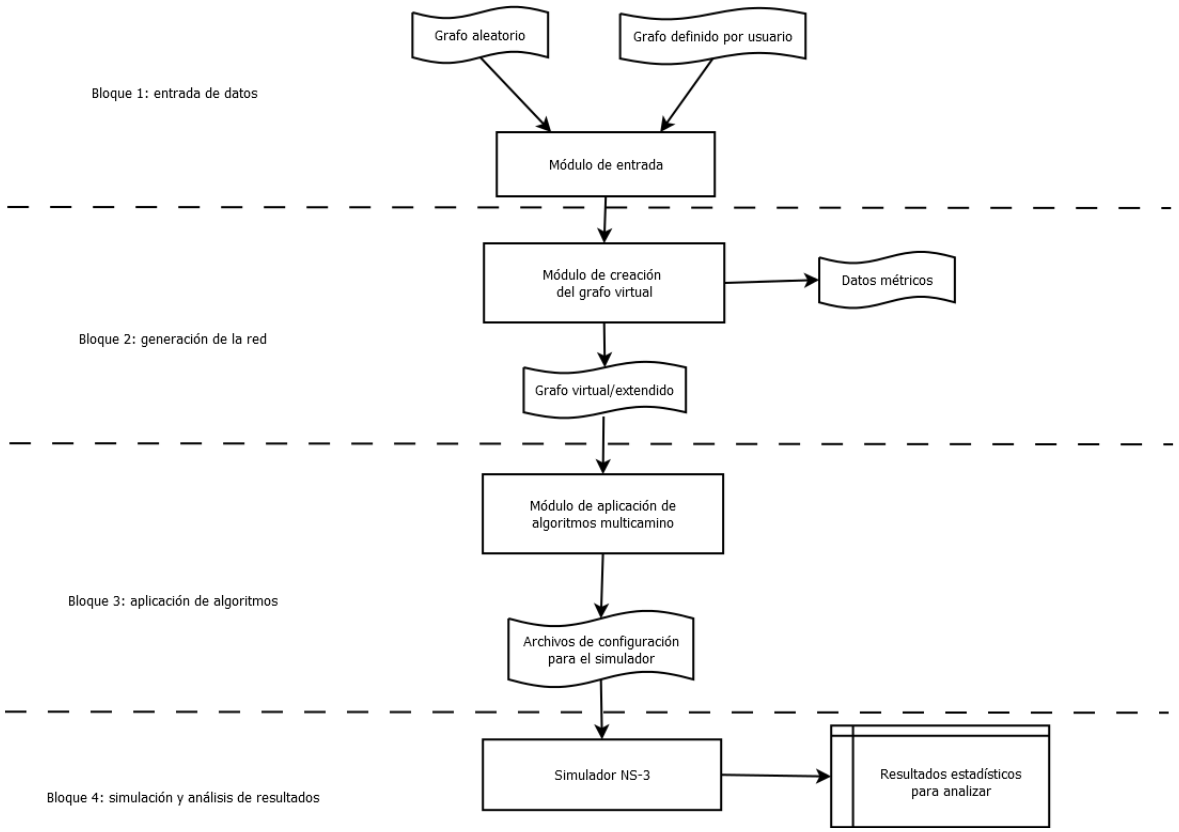



*Figura 1.1: Descripción del flujo de trabajo.*

# Capítulo 2: redes malladas inalámbricas de múltiples canales

Las redes multi-salto, habitualmente conocidas como redes *ad hoc,* han acaparado un gran interés por parte de la comunidad científica en los últimos años, especialmente el del grupo de trabajo MANET (*Mobile Ad hoc Networks*) del IETF. Dicho grupo ha trabajado en desarrollar una serie de técnicas de encaminamiento apropiadas para este tipo de topologías.

El interés en esta clase de redes ha evolucionado en los últimos tiempos: se ha pasado de un uso casi exclusivo de ellas en escenarios en los que una estructura cableada estática era inviable a pasar a ser un tipo de topología relevante en los sistemas inalámbricos de nueva generación, como las redes de comunicaciones móviles o las redes de sensores (WSN).

En este capítulo realizaremos un breve repaso de las características de las redes multi-salto, así como de los principales protocolos de encaminamiento que el grupo MANET ha desarrollado para ellas. A continuación presentaremos con más detalle el protocolo de balanceo de carga y enrutamiento LIBRA (*Load and Interference Balanced Routing Algorithm*) analizando por qué es adecuado para las redes multi-salto que incorporan varios canales inalámbricos; y emplearemos, apoyándonos en dicho protocolo, una técnica conocida como la modificación del grafo subyacente para transformar una red mallada multicanal en una red a la que poder aplicar algoritmos de encaminamiento clásicos.

## 2.1 Introducción a las redes multi-salto

Una red multi-salto, o red *ad hoc*, puede definirse como un conjunto de nodos autónomos que se agrupan de manera genérica sin que exista un elemento centralizador o estructura subyacente. En una red multi-salto, dichos nodos tienen la capacidad de moverse libremente y de forma arbitraria, lo que hace que la topología pueda variar impredeciblemente durante el tiempo de operación de la red. Reciben su nombre del hecho de que las rutas entre dos nodos cualesquiera de una red *ad hoc* pueden estar formadas por más de un salto inalámbrico.

Las características más destacables de estas redes son:

- Inalámbricas: los nodos se comunican a través de un canal compartido, el canal radio.
- Espontáneas: la topología de red se establece de forma dinámica y sin planificación.
- Autonomía de los nodos: las redes *ad hoc* no dependen de una estructura o de un coordinador de red, sino que cada elemento de la red genera y encamina su propio tráfico.
- Encaminamiento multi-salto: todos los nodos de la red participan en el enrutamiento de paquetes.
- Movilidad: la topología de red puede variar con el tiempo.

### Protocolos de encaminamiento multi-salto

Debido a la creciente popularidad de las redes ad hoc se han desarrollado en los últimos años un notable número de protocolos de encaminamiento que buscan adaptarse sus peculiares características. Podemos clasificar la mayoría de estos protocolos en dos grupos: preventivos y reactivos

Decimos que un protocolo es **preventivo** cuando los terminales disponen en todo momento de información completa sobre la topología de la red. Para ello, cuando se utiliza un protocolo preventivo, todos los nodos de la red realizan periódicamente un envío y procesado de mensajes de control. Esto presenta una gran ventaja: cuando un nodo quiera comunicarse con otro simplemente tendrá que mirar en sus propias tablas para encontrar una ruta hasta el mismo. También tienen una importante desventaja: el envío de mensajes de control genera un tráfico adicional en la red que supone un gasto de ancho de banda y energético.

Algunos de los protocolos preventivos más relevantes son:

- Vector-distancia: cada nodo almacena en su memoria información acerca de la distancia (número de saltos) hacia los posibles nodos, junto con el siguiente salto para alcanzarlos. Se envían mensajes broadcast cada vez que se detecta un cambio en un enlace para mantener las tablas de rutas actualizadas.

- Estado del enlace: cada nodo dispone de información sobre toda la topología de la red, incluyendo costes asociados a cada enlace. Cuando quiere enviar un paquete, ejecuta un algoritmo de búsqueda del camino mínimo. Como en vector-distancia, se mantienen las tablas actualizadas mediante mensajes broadcast cada vez que varía un enlace en la topología.

Estos dos protocolos tradicionales fueron optimizados más tarde por el grupo MANET del IETF. Dicho grupo lleva desde 1998 trabajando en el desarrollo de varios protocolos que se ajusten a las características de las redes multi-salto, soluciones de las cuales no todas han gozado de continuidad. Algunas de las que más relevancia han tenido son:

- *Optimized Link State Routing Protocol* (OLSR) es una optimización de los protocolos vector-distancia y estado del enlace adaptándose a las características intrínsecas de las redes *ad hoc*. Su principal mejora respecto a estos es el empleo de una monitorización distribuida en la que un conjunto de nodos conocido como MPR (*MultiPoint Relay*) se encarga de propagar a través de la topología los cambios que se producen en los enlaces. De esta forma se consigue reducir en gran porcentaje la sobrecarga por mensajes de control en la red.
- *Topology Broadcast based on Reverse-Path Forwarding* (TBRPF) también es una variante del protocolo de estado del enlace. Cada nodo construye un árbol representando la topología de la red, y obtiene la ruta hacia el resto de nodos aplicando un algoritmo de camino mínimo a dicho árbol (típicamente Dijkstra). Se diferencia de OLSR principalmente en que los cambios en la topología se notifican diferencialmente, comunicándose sólo a los vecinos del nodo que los origina.

Por otra parte tenemos los protocolos **reactivos**, en los cuales los terminales no mantienen información topológica acerca de la red. Cuando un nodo quiere comunicarse con otro, inicia un proceso de descubrimiento de ruta. Esto reduce en gran medida la cantidad de mensajes de señalización que atraviesan la red, aumentando el ancho de banda disponible para transmitir datos. El inconveniente que presentan es el retardo introducido debido al tiempo de búsqueda de la ruta cada vez que se quiere iniciar una comunicación.

Desde el comienzo de las actividades del grupo MANET fueron los protocolos reactivos los que mayor interés suscitaron dentro del grupo. Los más destacados dentro de esta familia de protocolos son:

- *Ad Hoc On-Demand Distance Vector Routing* (AODV). Aparece como una evolución del protocolo vector-distancia tradicional, en la que no es necesaria la propagación de mensajes de control ya que las rutas se construyen bajo demanda. Requiere de una fase de descubrimiento de ruta en la que un mensaje de *Route Request* (RREQ) se propaga por la red. Cuando alcanza al nodo objetivo, éste devuelve un *Route Reply* (RREP) que viaja hasta la fuente utilizando la ruta que siguió previamente el RREQ.
- *Dynamic Source Routing for Mobile Ad Hoc Networks (*DSR*)* comparte con AODV las bases de su funcionamiento (encaminamiento en la fuente). Se diferencia de éste en que cada datagrama enviado por el nodo fuente incluye la ruta completa; así, los nodos intermedios sólo tienen que extraer el próximo salto para reenviarlo.
- *Dynamic MANET On-Demand Routing* (DYMO) es una evolución de AODV que busca reducir su complejidad y aumentar al mismo tiempo su flexibilidad. Para ello se ajusta el formato de los mensajes de control, reduciendo la sobrecarga en la red, y se elimina el uso de mensajes *Hello.*

Todos estos protocolos, como hemos visto, emplean algoritmos de encaminamiento simples para establecer sus rutas, ya sea mediante Dijkstra o basándose en inundación. De esta forma, se obtienen rutas simples entre cada par de nodos de la red. Sin embargo, la naturaleza multi-canal de este tipo de redes nos plantea una nueva posibilidad: aprovechar esos canales para dividir el tráfico entre dos nodos por dos o más rutas.

Pero realizar esa aproximación no es ni mucho menos trivial: la naturaleza inalámbrica del canal puede provocar interferencias entre dichas rutas con la consiguiente aparición de errores y pérdida de paquetes. Por ello, es necesario desarrollar métricas de encaminamiento que tengan en cuenta esta característica e incluirlas dentro de nuestro protocolo de enrutamiento.

## 2.2 Redes malladas multicanal: MIC y LIBRA

Las características únicas de las redes malladas, como son las posiciones dinámicas de los nodos y la naturaleza compartida del medio inalámbrico, invalidan las soluciones existentes tanto para redes cableadas como inalámbricas e introducen nuevos desafíos para balancear la carga de la red. Este balanceo de carga es necesario en cualquier tipo de red, para evitar áreas de congestión e incrementar la utilización de la red; ya que en una red estática podrían existir rutas mal planteadas durante largos periodos de tiempo, resultando en congestión y en un uso ineficiente de los recursos. A pesar de que el balanceo de carga ha sido estudiado ya en redes cableadas, los canales de comunicación inalámbricos en redes malladas introducen nuevos desafíos:

- Cada uno de los nodos de una red inalámbrica puede estar equipado con varias interfaces radio, a lo que llamaremos nodo multicanal; y cada una de esas interfaces puede ser configurada para operar en un canal diferente, lo que permitirá mejorar la capacidad de la red. Todos los nodos inalámbricos deben enrutar el tráfico proveniente del resto hacia un punto de acceso a internet (un *gateway*). Por tanto, el objetivo principal de los protocolos de enrutamiento debe ser equilibrar la carga entre todos los nodos de la red.

- El principal motivo para que las soluciones para redes cableadas no sean válidas en redes inalámbricas es la naturaleza compartida del canal. Al transmitir todos los nodos en el mismo canal inalámbrico, tanto la interferencia *inter-flow* (entre nodos que llevan flujos de datos diferentes) como la *intra-flow* (entre nodos que llevan el mismo flujo de datos) puede afectar a la carga de tráfico de los nodos.

- Al ser estos protocolos existentes poco eficientes para ejecutarse en redes malladas, en los últimos años han comenzado a emerger protocolos de balanceo de carga específicos para redes malladas, como ETX, ETT o WCETT. A diferencia de los protocolos bajo demanda de las redes *ad hoc*, estos nuevos protocolos se basan en enrutamiento proactivo para adecuarse a la característica estática de este tipo de redes. Esto es, envían mensajes de actualización periódicamente pero con una frecuencia baja, o bien cuando la topología de la red varía. Sin embargo, ninguno de ellos captura los mencionados problemas de *inter-flow* e *intra-flow.*

**Supuestos iniciales**
A la hora de diseñar un protocolo que realice el balanceo de carga de forma óptima en una red mallada inalámbrica, la primera premisa que asumimos es que cada nodo está equipado con una o más interfaces radio, y cada interfaz está preconfigurada a cierto canal; además, las interfaces configuradas a diferentes canales no interfieren entre sí. Partiendo de ese supuesto, es importante señalar que si un nodo *i* se comunica con un vecino usando la interfaz radio *r* en el canal *c*, el nodo *i* no podrá comunicarse concurrentemente con otros vecinos usando ese mismo canal *c* ya sea para recibir o transmitir. Por tanto, el objetivo de balanceo de carga deberá ser definido en función de la utilización de canales inalámbricos, en lugar de enlaces inalámbricos.

**Tipos de protocolos de enrutamiento**
Debido a la extrema dificultad de realizar un balanceo de la carga óptimo en redes malladas, es necesario diseñar protocolos de enrutamiento heurísticos con el objetivo de aproximarnos a esa dicha solución óptima. La clave para que un protocolo heurístico nos de un buen rendimiento es el diseño de su función de *path weight* (métrica de enrutamiento).

Dependiendo de cómo se enrutan los paquetes, los protocolos de enrutamiento en redes malladas pueden dividirse en dos categorías:

1. Enrutamiento en la fuente, como *Link Quality Source Routing* o *Multi-hop Communication Routing*: incluyen el camino a seguir por los paquetes íntegro en sus cabeceras, de forma que los nodos intermedios sólo tienen que reenviar los paquetes siguiendo dicho camino. Debido a que los paquetes en redes malladas podrían ser muy pequeños, esto genera una sobrecarga que no es interesante.
2. Enrutamiento salto a salto: el origen sólo pone la dirección final y los nodos van redirigiendo el paquete según su tabla de enrutamiento para llegar a dicho destino. Esto genera un *overhead* mucho más bajo, lo cual hace de este tipo de enrutamiento el óptimo para las redes malladas.

**Requisitos de los protocolos de enrutamiento**
Para asegurar un rendimiento óptimo de los protocolos heurísticos de enrutamiento, sus funciones path weight deben cumplir 4 requisitos:

a. Función de *Path Weigh*t estable: las variaciones en el tráfico en enlaces en una red mallada pueden ser amplias y a intervalos irregulares, haciendo muy difícil usar funciones de *path weight* sensibles a la carga de los enlaces. Sin embargo, funciones que dependan de la topología (número de saltos, capacidad del enlace), son más estables y preferibles.

b. Capturar las características de la red: como ya hemos apuntado, en las redes malladas inalámbricas existen dos tipos de interferencias que las funciones de *path weight* deberán reflejar. La primera es la interferencia *intra-flow*, que se refiere al hecho de que nodos en el camino de un mismo flujo compiten por el ancho de banda del canal si transmiten por el mismo canal. Por contra, la interferencia *inter-flow* refleja que nodos que lleven diferentes flujos compiten por el ancho de banda si transmiten utilizando a su vez el mismo canal.

c. Cálculo de los caminos de coste mínimo e isotonicidad: una vez hayamos definido métricas para el grafo, tendremos que aplicarle algoritmos de búsqueda

de camino mínimo basados en los algoritmos clásicos (Bellman-Ford o, en nuestro caso, Dijkstra). Para asegurar que dichos algoritmos pueden encontrar caminos mínimos se tiene que cumplir una propiedad fundamental: la isotonicidad. La isotonicidad es una condición suficiente y necesaria para que tanto el algoritmo de Bellman-Ford como el de Dijkstra encuentren caminos de coste mínimo en un grafo.

Decimos que una función *W(.)* es isotónica si

$$W(a) \leq W(b)$$

implica tanto que

$$W(a + c) \leq W(b + c)$$

como que

$$W(c' + a) \leq W(c' + b)$$

para cualesquiera valores de *a,b,c,c'*. (2.1)

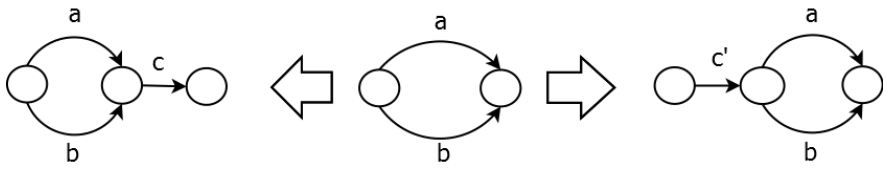



*Figura 2.1: Concepto de isotonicidad.*

d. Enrutamiento libre de ciclos: si utilizásemos el enrutamiento en fuente, utilizar los algoritmos clásicos de enrutamiento nos proporcionaría rutas libres de ciclos, ya que el nodo fuente tiene control completo sobre el camino de los flujos. Sin embargo, en enrutamiento salto a salto podrían aparecer ciclos a menos que la función de coste sea isotónica.

Por tanto, se deduce que para asegurar el coste mínimo y la ausencia de ciclos en el enrutamiento en redes malladas es muy importante comprobar la isotonicidad de las métricas de enrutamiento.

### 2.2.1 Definición de las funciones de *path weight*

A continuación se presenta una función de *path weight* dependiente de la topología de red llamada métrica de interferencia y cambio de canal (Metric of Interference and Channel Switching, MIC), compuesta a su vez por dos métricas: Uso de Recursos dependiente de la Interferencia (Interference-aware Resource Usage, IRU) y Coste del Cambio de Canal (Channel Switching Cost, CSC). Estas dos métricas de MIC representan diferentes características de las redes malladas. La métrica IRU captura los efectos de la interferencia *inter-flow* y las diferencias entre las velocidades de transmisión y las tasas de pérdida de los enlaces inalámbricos; mientras que la métrica CSC captura el impacto de la interferencia *intra-flow*.

**Conceptos de Inter-flow e Intra-flow**

Hemos mencionado ya en múltiples ocasiones la existencia de dos tipos de interferencia que afectan a las redes inalámbricas malladas: inter e intra flow. Pero, ¿a qué nos referimos exactamente cuando nombramos estos dos problemas?

Las redes inalámbricas presentan una diferencia fundamental con las cableadas, y es que su medio de transmisión es el espectro electromagnético. En el caso concreto de 802.11, se utiliza la banda de 2.4 GHz, dividida en 14 canales parcialmente solapados. En concreto, dos canales se solapan si están separados entre sí por menos de cuatro canales; es decir, si queremos definir canales completamente ortogonales tendremos solamente tres canales: 1, 6 y 11.

Esta característica compartida del espectro plantea una serie de problemas en redes malladas, en las cuales no existe un Punto de Acceso (AP) el cual podría coordinar las transmisiones para que los nodos que emplean el mismo canal no se interfieran entre sí.

El primer problema (y el más complicado de subsanar) es la interferencia entre flujos de datos diferentes que utilizan el mismo canal de transmisión (***Inter-flow***), como vemos en la figura 2.2.

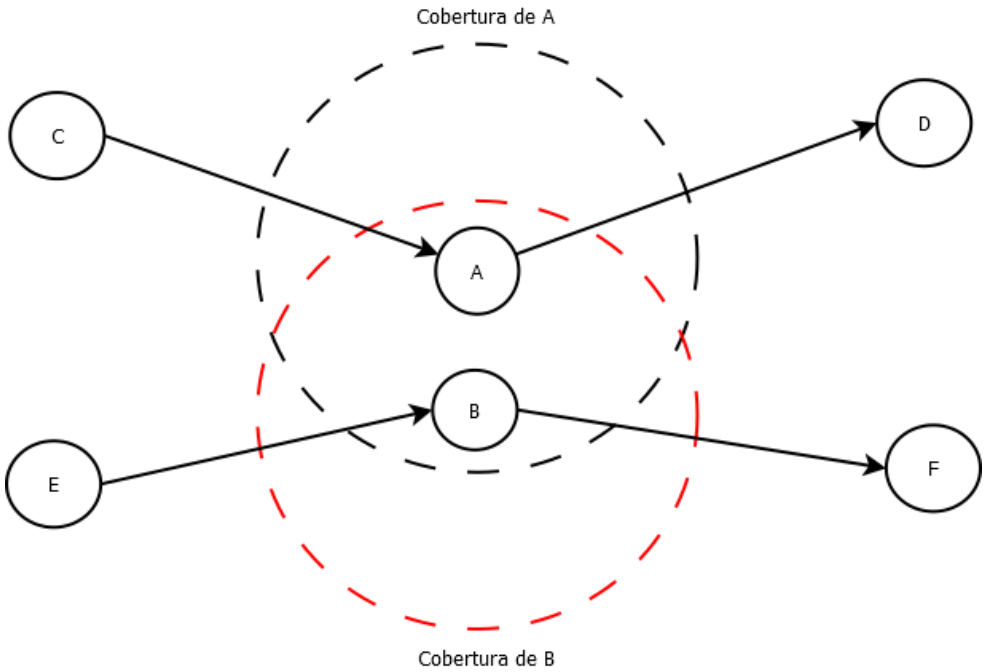



*Figura 2.2: Inter-flow entre dos nodos próximos.*

Como podemos apreciar, los radios de cobertura de A y B están superpuestos. En el caso de que ambos utilicen el mismo canal para transmitir, podría haber interferencias que resultasen en la pérdida o transmisión errónea de paquetes. Para evitar esto en nodos con múltiples interfaces es necesario elaborar un protocolo de enrutamiento en el que cada nodo tenga en cuenta los canales utilizados todos los demás, con el fin de evitar utilizar los canales más comúnmente empleados por estos.

El otro tipo de dificultad que nos encontramos en las redes inalámbricas es el uso del mismo canal para transmitir dentro del mismo flujo (***intra-flow***). Si un nodo B recibe datos de A a través del canal X, no podrá utilizar ese mismo canal para transmitir datos. Y esto no ocurre sólo con el salto inmediatamente anterior. Si los radios de cobertura de los nodos se superponen (figura 2.3), también el salto previo ha de ser tenido en cuenta a la hora de elegir el canal para transmitir.

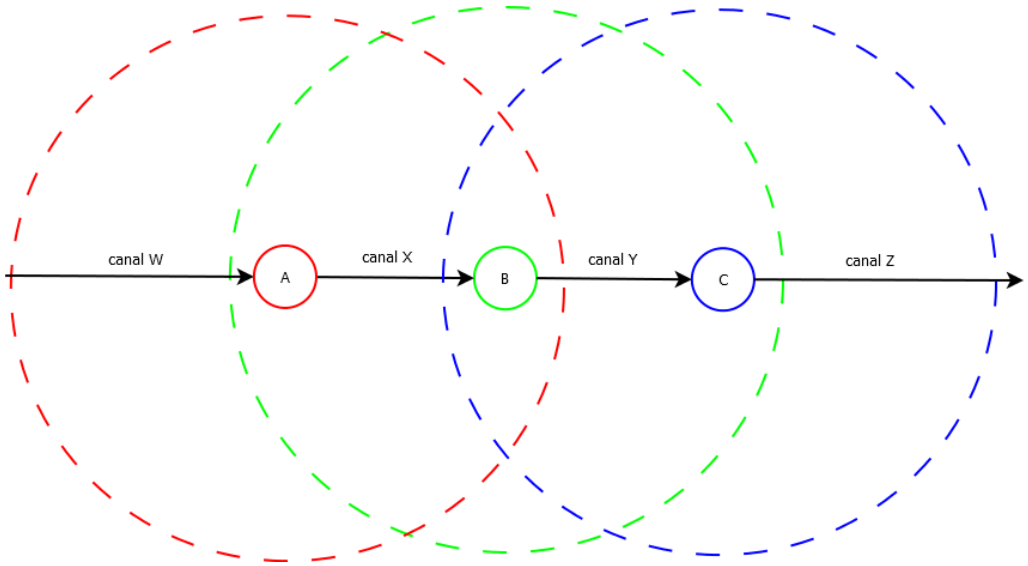



*Figura 2.3: Intra-flow dentro de un flujo de datos.*

En este caso, nos aseguraremos de que el nodo C conoce tanto el canal Y que B ha utilizado para transmitirle el paquete como el canal X que A ha empleado para transmitírselo a B. De este modo asignaremos pesos a los posibles valores que pueda tomar el canal Z.

Una vez señalados los dos problemas más importantes que nos presenta una topología inalámbrica multicanal, veamos cómo modelar estas dificultades para diseñar un enrutamiento eficiente.

**Interference-aware Resource Usage (IRU)**
Para capturar las diferencias en la velocidad de transmisión y tasa de pérdidas de los canales inalámbricos así como la interferencia inter-flow, la métrica IRU se define como:

$$IRU_{ij}(c) = ETX_{ij}(c) * |N_i(c) \cup N_j(c)| \qquad (2.2)$$

donde *Ni(c)* es el conjunto de vecinos del nodo con los que el nodo *i* interfiere cuando transmite en el canal c, y *|Ni(c) u Nj(c)|* es, por tanto, el número total de nodos que podrían interferir en la transmisión entre *i* y *j* a través del canal *c*. El término *|Ni(c) u Nj(c)|* se introduce para capturar el inter-flow, ya que si un flujo está transmitiendo un paquete en el enlace (*i,j*) por el canal *c*, ningún nodo vecino podrá usar dicho canal para transmitir.

El término *ETXij(c)* se refiere a la métrica ETX (*Expected Transmission Count*), la cual hace que la métrica IRU sea dependiente de la topología de la red. Asigna a un enlace (*i,j*) un coste inversamente proporcional a la distancia entre los nodos *i* y *j*, siempre que

esta distancia sea menor al radio de cobertura definido para los nodos. La distancia entre los nodos se calcula según sus posiciones en el espacio euclídeo:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (2.3)$$

Una vez conocida la distancia entre nodos, la calidad del enlace será inversamente proporcional a ésta, y sólo existirá enlace en el intervalo (0, *dmax*).

**Channel Switching Cost (CSC)**

La métrica IRU no es capaz de explotar las capacidades multicanal de los nodos para reducir la interferencia intra-flow. Entre dos caminos que tienen el mismo peso IRU, la diversificación del camino (es decir, que los nodos empleen canales distintos en cada salto) proporciona una interferencia intra-flow mucho menor que usar el mismo canal durante todo el flujo. Para capturar esto utilizaremos la métrica del Coste de Cambio de Canal.

La métrica CSC se basa en que para eliminar la interferencia intra-flow entre un nodo *X* y su salto previo, *prev(x)* el nodo *X* debe transmitir al salto siguiente utilizando un canal diferente al que usó su predecesor. Es decir:

$$CSC_x = w1 \; si \; CH\big(prev(x)\big) \neq CH(x)$$
$$CSC_x = w1 \; si \; CH\big(prev(x)\big) = CH(x)$$
$$0 \leq w1 < w2 \qquad (2.4)$$

La relación *w2>w1* captura el hecho de que, debido a la interferencia intra-flow, usar el mismo canal en el nodo *X* y en *prev(x)* impone un coste más alto que utilizar canales diferentes. w1 captura el coste de utilizar múltiples radios simultáneamente. Normalmente estos costes son 0, pero en las ciertas redes malladas (WSN) los requisitos energéticos son tan restrictivos que a veces estas operaciones no deben ser ignoradas, utilizándose en ese caso w1 para representar dichos costes.

Con esta expresión tendríamos el concepto básico de la métrica CSC. Sin embargo, en ese caso sólo estamos considerando la interferencia entre dos nodos consecutivos de un camino. Dependiendo del rango de cobertura de los nodos, la interferencia intra-flow también podría existir entre nodos que están más alejados. Suponiendo que dicha interferencia se dé entre nodos que están a una distancia de dos saltos la nueva métrica CSC será:

$$CSCx = \begin{cases} w2 & si \;\; CH(prev2(x)) \neq CH(x) = CH(prev(x)) \\ w3 & si \;\; CH(prev2(x)) = CH(x) \neq CH(prev(x)) \\ w2 + w3 & si \;\; CH\big(prev2(x)\big) = CH(x) = CH(prev(x)) \\ w1 & en\ otro\ caso \end{cases}$$

$$0 \leq w1 \ll w3 < w2 \qquad (2.5)$$

Donde *w3* captura la interferencia intra-flow entre los nodos *prev2(x)* y *X*, y *w2* captura la interferencia entre los nodos *prev(x)* y *X*. La relación *w3<w2* responde a que a mayor distancia entre nodos, menor interferencia existirá entre ellos.

**Metric of Interference and Channel-switching (MIC)**
Para capturar todas las características de las redes malladas, combinaremos las métricas IRU y CSC para crear una nueva función de *path weight* conocida como métrica de interferencia y cambio de canal (MIC):

$$MIC(p) = \alpha * \sum_{link\ l \in P} IRU(l) + \sum_{nodo\ i \in P} CSC_i \qquad (2.6)$$

donde *P* es un camino en la red. En esencia, el componente IRU en la función MIC captura la utilización del ancho de banda total disponible en la red por parte de un flujo a lo largo de *P* y tiene como objetivo disminuir la utilización del canal. El componente CSC representa el rendimiento de los flujos encaminados por p y tiene como objetivo reducir el retardo e incrementar el *throughput* de cada flujo individual. Estos dos objetivos podrían entrar en conflicto entre ellos, por ejemplo:

Supongamos que existen dos caminos *p1* y *p2* entre un par de nodos. *p1* tiene un mayor número de canales y por tanto mayor throughput, mientras que *p2* consume menos tiempo en el canal inalámbrico, al atravesar enlaces de mayor ancho de banda (aunque estos enlaces utilicen todos el mismo canal). El factor se introduce para representar el compromiso entre estos dos beneficios. Un valor alto significa que balancear la carga en la red es más importante que obtener buen rendimiento por flujo, mientras que un más pequeño dará mayor importancia al rendimiento de cada flujo individual, incluso si la carga de tráfico no está perfectamente balanceada, y algunos nodos vecinos reciben pocos recursos debido a la interferencia inter-flow.

Una vez hemos elegido la métrica a utilizar, el siguiente paso será adaptar la topología mallada a esta nueva definición, en lo que denominaremos la creación de la Red Virtual.

### 2.2.2. Descomposición de MIC en una Red Virtual: LIBRA

Mapeando cuidadosamente una red real en una virtual, podemos descomponer la métrica MIC en una serie de asignaciones de pesos isotónicas dentro de la red. El objetivo de esta descomposición es asegurar que los algoritmos de encaminamiento que emplearemos en el siguiente capítulo pueden encontrar caminos mínimos sin crear ciclos. El nuevo esquema de enrutamiento que generaremos se conoce como LIBRA: *Load and Interference Balanced Routing Algorithm*.

Pero, ¿por qué MIC no es isotónica? Pongamos como ejemplo el siguiente escenario:

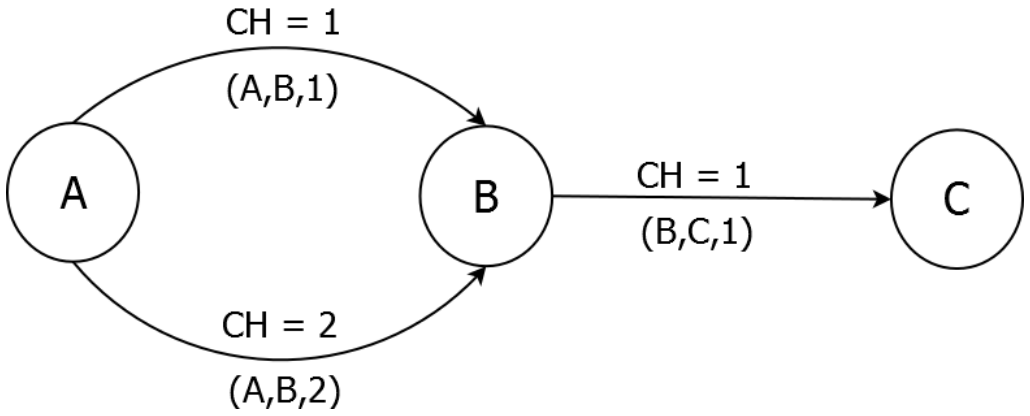



*Figura 2.4: Isotonicidad en MIC.*

En este ejemplo, el coste adicional que el enlace *(B,C,1)* añade al camino no sólo depende de su propio estado, sino que también está relacionado con el canal que se asigna al enlace que lo precede. Debido al canal común utilizado por los enlaces *(A,B,1)* y *(B,C,1)*, añadir el enlace *(B,C,1)* al camino *(A,B,1)* introduce un coste más alto que añadírselo al camino *(A,B,2)*. Por tanto, no se cumple la definición de isotonicidad (fig. 2.1).

Para enfrentarnos a este problema, tenemos que darnos cuenta de que para MIC el hecho de que el incremento del coste sea diferente al añadir un enlace sólo está relacionado con la asignación del canal del salto previo en el camino. Esta asignación está limitada al número de interfaces radio que el nodo tiene, ya que cada interfaz está configurada en un canal. Por tanto, introduciendo varios nodos virtuales que representen estas posibles asignaciones de canal para el enlace previo podemos convertir MIC para conseguir asignaciones de coste del enlace isotónicas. En una primera aproximación, para cada canal *c* al que estén configuradas las interfaces radio de un nodo *X* introduciremos dos nodos virtuales, *Xi(c)* y *Xe(c)*. *Xi(c)* representa que el nodo *prev(x)* transmite al nodo *X* por el canal *c*. *Xe(c)* representa que el nodo *X* transmite a su siguiente salto por el canal *c*. El subíndice *i* indica “ingress” y el subíndice *e* significa “egress”.

Además, introduciremos dos nodos: *X*- y *X*+, a los cuales conoceremos como el nodo less y plus, respectivamente. Dichos nodos representan el hecho de que el nodo *X* sea el destino o fuente de un flujo, y por tanto no tendrá coste alguno enviar o recibir datos desde ellos.

En el esquema siguiente podemos ver esta primera aproximación para un nodo Y que utiliza 2 canales, 1 y 2.

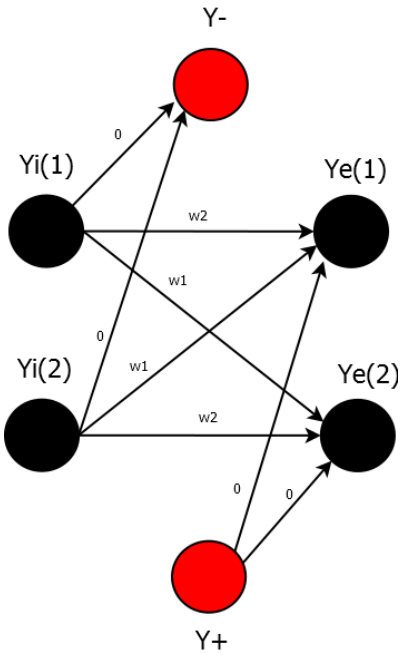



*Figura 2.5: primera aproximación a un nodo virtualizado.*

Esta primera descomposición nos sirve para entender la necesidad de crear nodos ingress y egress para cada canal. Sin embargo, como vimos en la definición de la métrica CSC, para redes malladas inalámbricas es interesante tener en cuenta no sólo los canales utilizados por el salto inmediatamente anterior, sino también los que incorpora el nodo que está dos saltos más atrás en el camino (*prev2(x)*). Así, es necesario extender la virtualización ya realizada.

Consideremos un nodo *X* con *m* radios, cada una configurada a un canal diferente. Por cada canal *C* en el nodo *X*, se introduce *m+1* nodos tipo egress *Xe(Ca,C)*. *Xe(Ca,C)* representa que un flujo llega al nodo *X* desde el canal *Ca* y se dirige al nodo siguiente a través del canal *C*. *Ca* puede tener dos tipos de valores: bien uno de los *m* canales configurados en el nodo *X* o bien -1 para representar que el flujo es generado por el mismo nodo *X*, por lo que no hay ningún *prev2(x)*.

Además, introduciremos otros *m+1* nodos ingress virtuales *Xi(Cp, c)* por cada canal *c* que implementa el nodo *X*. *Xi(Cp, c)* representa que un flujo viaja desde *prev2(x)* hasta *prev(x)* por el canal *Cp* y llega al nodo *X* a través del canal *c*. *Cp* tiene, igual que en el caso de los nodos egress, dos tipos de valores posibles: bien uno de los m canales configurados en el nodo *X* o bien -1 para representar que *prev2(x)* utiliza un canal no implementado por el nodo *X*.

Veamos ahora cómo quedaría el nodo Y:

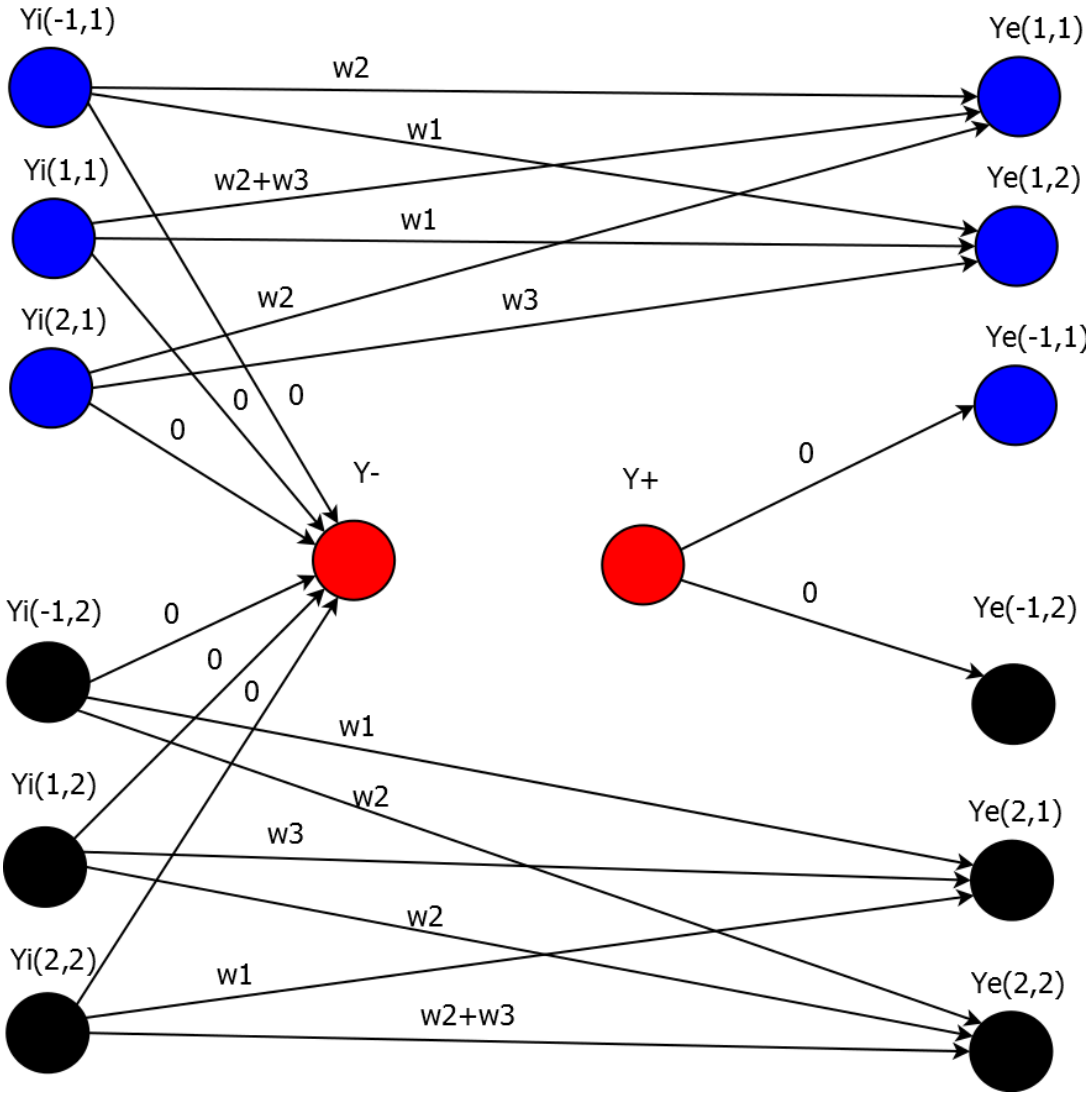



*Figura 2.6: nodo virtualizado completo.*

Llegados a este punto, hemos conseguido simular mediante la creación de nodos virtuales la métrica CSC. Por tanto, para completar la métrica MIC restará añadir la interferencia inter-flow mediante la métrica IRU.

En el siguiente esquema podemos ver la representación de la virtualización completada de una red para reflejar la métrica MIC.

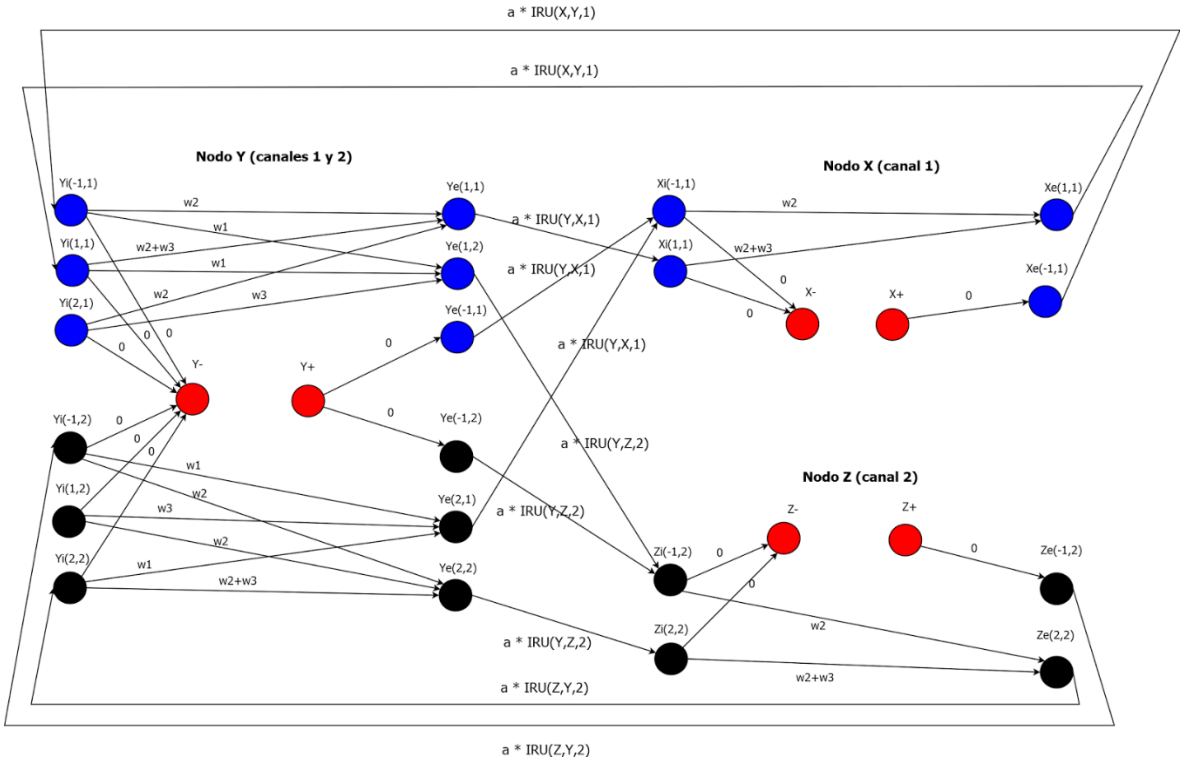



*Figura 2.7: red de tres nodos y tres enlaces transformada.*

Teniendo en cuenta que la red original sólamente tenía 3 nodos de los cuales 1 empleaba dos interfaces y el resto una, queda claro que el proceso de convertir la red en isotónica da como resultado redes mucho más complejas, con mayor número de nodos y enlaces.

El número total de nodos necesarios para representar un nodo real multi-interfaz queda definido así:

1. *C* nodos (donde *C* es el número de canales que implementa el nodo del grafo real) para los flujos que vienen de canales desconocidos: *Xi(-1,C)*; y otros *C* para los flujos que salen del propio nodo por cada canal *Xe(-1,C*).
   Total: *2*C* nodos.

2. Para todos los canales implementados por el nodo tendremos todos los posibles pares *Xe(C,C)* y *Xi(C,C)*. Total: $2 * C^2$ nodos.

3. Un nodo Plus y un nodo Less.

$$Total\ nodos\ virtuales\ /\ nodo\ real\colon 2 * C^2 + 2 * C + 2 \qquad (2.6)$$

Mediante esta expresión podemos calcular el número máximo de nodos que tendrá nuestro grafo virtual, lo cual será útil para limitar el tamaño de las estructuras que manejaremos.

$$Total\ nodos\ virtuales = \ nodos\ reales * (2 * C^2 + 2 * C + 2) \qquad (2.7)$$

## 2.3 Implementación del protocolo

A continuación explicaremos brevemente cómo se implementó todo el proceso de transformación de la red mediante C++.

Mediante un fichero de entrada o bien aleatoriamente, generamos una estructura de entrada *nodo* que recoge la posición en el plano (x,y) de los nodos, así como si incorporan o no los canales presentes en el medio (hasta 4 inicialmente). Nos apoyamos para ello en las clases *Leer* y *randomGen*.

Esa estructura es recogida por la clase Virtual.cpp, que incluye los métodos necesarios para generar la red virtual:

- El método *NodosVirtuales* genera la estructura en la que guardaremos toda la información del grafo, en este caso un array de listas enlazadas.

- Los métodos *PlusyLess* y *CanalVirtual* introducen en la estructura la información acerca de cada nodo del grafo: el tipo de cada nodo (plus, less, ingress, egress), su canal utilizado y los predecesores de primer y segundo orden.

- El método *ListaEnlazada* genera los enlaces entre todos los nodos de la red, asignándole los costes según la métrica MIC. Dicha métrica se define en los métodos *Iru* y *Etx*, que la generan en función de la topología de la red.

- Una vez generado el grafo, se envía a un fichero de salida en el que se incluye cada nodo con sus enlaces y costes.

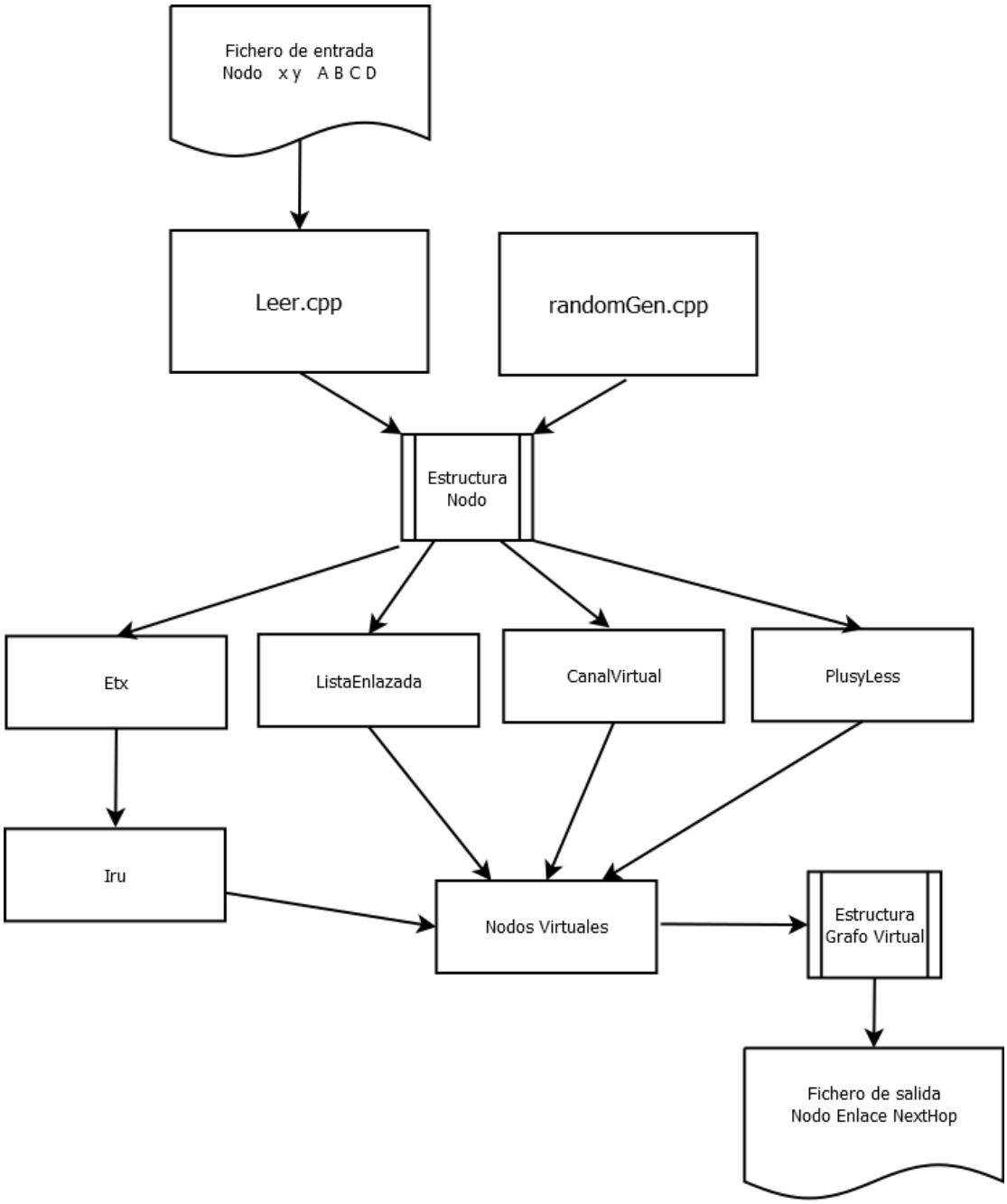



*Figura 2.8: Esquema de la implementación de LIBRA.*

## Conclusiones

A lo largo de este capítulo hemos realizado un análisis de la situación actual de las redes inalámbricas multi-salto, enumerando y describiendo brevemente los protocolos que el grupo de investigación MANET ha desarrollado con el objetivo de proponer una solución al encaminamiento en esta clase de tecnología.

Hemos hecho hincapié también en las características de este tipo de redes, centrándonos principalmente en los desafíos que presenta la naturaleza compartida del canal inalámbrico cuando se combina con la ausencia de un agente controlador de la red.

El protocolo LIBRA, junto con la métrica MIC, supone un gran paso para buscar nuevas soluciones que reflejen mejor el característico comportamiento de este tipo de redes. LIBRA nos permite, por medio de la transformación de las interfaces en nodos simples, convertir una red multicanal (no isotónica) en un grafo sobre el que podremos aplicar protocolos de encaminamiento tradicionales, si bien lo hace a costa de generar una red de mayor tamaño en cuando a número de nodos y enlaces.

Por su parte, MIC refleja los problemas de *inter-flow* e *intra-flow* que aparecen en las redes inalámbricas malladas, asegurándonos que obtendremos un rendimiento satisfactorio al emplear varios canales. Además, tenemos la garantía de que la nueva red cumple el criterio de isotonicidad, de forma que los algoritmos de búsqueda de camino clásicos como Dijkstra o Bellman-Ford serán capaces de cumplir su función.

Esto último sera crucial en el siguiente capítulo de este trabajo, puesto que los algoritmos que emplearemos para enrutar paquetes en el procolo MPTCP (Multi-Path TCP) están todos basados en el algoritmo de Dijkstra.

# 3. Algoritmos Multicamino. El protocolo MPTCP.

## 3.1 Introducción

Desde sus orígenes, Internet ha visto como uno de sus protocolos, TCP y sus sucesivas versiones, se ha convertido en la solución de capa de transporte más extendida superando a otras alternativas como UDP, SCTP (Stream Control Transmission Protocol) o DCCP (Data Congestion Control Protocol). Sin embargo, en la actualidad las características de TCP han dejado de ser óptimas. Cuando el protocolo fue diseñado, la tecnología predominante eran los dispositivos y redes basadas en sistemas de una sóla interfaz (redes Ethernet mayoritariamente). Sin embargo, con el tiempo ese hecho ha cambiado y actualmente la mayoría de comunicaciones no corresponden con ese modelo. Por ejemplo, existen muchos servidores y centros de datos *multihost* que soportan topologías redundantes, permitiendo su acceso a través de varios caminos de forma simultánea.

Pero sin embargo el principal motivo del crecimiento de los sistemas de múltiples interfaces es el auge de los dispositivos móviles, cuyo impacto ha cambiado los arquetipos de las redes de acceso. Este tipo de dispositivos cuenta con múltiples interfaces que pertenecen a diferentes tecnologías de acceso radio (RATs). Un ejemplo claro es un dispositivo móvil de última generación, que cuenta tanto con un sistema 3G/4G como con una tarjeta de red WiFi. Combinar ambos sistemas de comunicaciones para conseguir una conexión en la que obtengamos un flujo de datos agregado con mayor *throughput* para una sóla aplicación se presenta como una solución de gran utilidad.

La comunidad investigadora ha realizado en los últimos años un gran esfuerzo en buscar una forma apropiada de dar soporte a conexiones fiables aprovechándose de esta nueva característica multi-interfaz. Uno de los resultados más importantes de estos estudios es el protocolo *Multi-Path TCP* (MPTCP), que puede verse como una evolución natural de TCP, que permite el uso efectivo y simultáneo de varios caminos (a los que a partir de ahora llamaremos subflujos) para llevar una sóla conexión de transporte. Sus mecanismos de control distribuyen la carga de datos disponible entre los subflujos activos, los cuales están relacionados con la dirección IP activa de la entidad. Mediante esta modificación de TCP, una aplicación puede beneficiarse de un throughput más alto y de una mayor robustez frente a errores.

Tradicionalmente, la mayoría de análisis de redes multi-camino estaban centrados en topologías cableadas donde la teoría de grafos se aplica de una forma más directa y los enlaces entre nodos son completamente controlables. Sin embargo, cada vez es más común encontrar estudios que se centran en el análisis del comportamiento de estos algoritmos en redes inalámbricas, puesto que la ganancia de robustez frente a errores que ofrecen estas soluciones es una figura de mérito, especialmente para escenarios en los que un enlace de poca calidad puede significar un gran incremento en la tasa de pérdida de paquetes.

MultiPath Transfer Control Protocol proporciona la capacidad de utilizar simultáneamente varios caminos (paths) entre dos extremos. Este protocolo ofrece el mismo tipo de servicio a las aplicaciones que TCP (es decir, transporte de un stream de datos fiable) y además añade los componentes necesarios para establecer y usar múltiples flujos TCP a través de caminos potencialmente disjuntos.

## 3.2 El protocolo MPTCP

MPTCP se define originalmente como una versión modificada del popular protocolo de capa de transporte TCP. Da soporte a un conjunto de extensiones que permiten a un terminal dividir una sola sesión en múltiples *subflujos* a través de la red para que transmiten simultáneamente la información. El protocolo se encargará de distribuir la carga entre esas sub-conexiones, las cuales atravesarán caminos potencialmente disjuntos, incluso usando diferentes tecnologías. MPTCP se confecciona para funcionar con estrategias multi-camino extremo a extremo, en las que uno o más de los terminales deben tener más de una dirección IP válida.

Su principio básico es simple: si un terminal posee múltiples puntos de conexión a la red (interfaces) ese hecho se puede explotar dividiendo el tráfico simultáneamente entre varias conexiones. Gracias a esta estrategia multi-camino, el rendimiento y la robustez de la comunicación mejoran en gran medida.

### 3.2.1 Consideraciones de diseño

Las dos restricciones que el IETF marca para limitar el alcance de MPTCP son:

- Debe ser compatible con la versión actual de TCP para facilitar su Para ello,, se establece que cualquier implementación de MPTCP tiene que ser capaz de dar soporte a cualquier aplicación que no esté diseñada específicamente para MPTCP. En dichos casos, los servicios no tendrán que ser capaces de diferenciar entre conexiones MPTCP y TCP en lo que a nivel de transporte se refiere. Para dichos servicios MPTCP tendrá que operar exactamente como TCP.

- Se asume de partida que uno o ambos hosts que van a emplear el protocolo tienen múltiples interfaces, y por tanto múltiples direcciones.

Para simplificar el diseño, se asume que la presencia de múltiples direcciones en un host es suficiente para indicar la existencia de múltiples caminos. Dichos caminos pueden no ser completamente disjuntos: podrían compartir uno o más routers entre ellos. Incluso en esa situación, el uso de múltiples caminos es beneficioso, mejorando el uso de recursos y la resistencia ante posibles fallos en los nodos. Los mecanismos de control de congestión empleados por MPTCP aseguran que este hecho no repercuta negativamente en el rendimiento del protocolo.

### 3.2.2 Objetivos de MPTCP

Los objetivos que el IETF se marca con la propuesta de MPTCP para que represente una alternativa apropiada al TCP original son:

1. Mejorar el *throughput*: el rendimiento que un flujo multi-camino ofrezca debe ser al menos tan alto como el de un flujo TCP clásico sobre el mejor camino posible.

2. No perjudicar: un subflujo MPTCP no puede requerir más recursos en los terminales extremo que los que requiere un flujo TCP que utiliza un sólo camino.

3. Equilibrar la congestión: ante una situación de congestión, MPTCP trasladará tanto tráfico como sea posible de los subflujos más congestionados hacia los menos congestionados.

### 3.2.3 MPTCP en la pila de protocolos

MPTCP opera en la capa de transporte, y trata de ser transparente tanto a la capa superior como a la inferior. Su funcionamiento en la pila se estructura en dos partes:

- La subcapa superior se encarga de las tareas orientadas a la aplicación, como son la iniciación o finalización de sesiones, el establecimiento de subflujos o la detección y uso de múltiples caminos.

- La subcapa inferior se encarga de aspectos de la red, gestionando cada uno de los subflujos creados durante la fase de establecimiento de conexión. Para ello, se crean tantas entidades como subflujos existen en la conexión.

Por debajo de estas entidades se asociará una instancia IP a cada uno de los subflujos, de forma que la operación de MPTCP sea transparente para el nivel de red, el cual lo tratará como varios clientes a nivel de transporte.

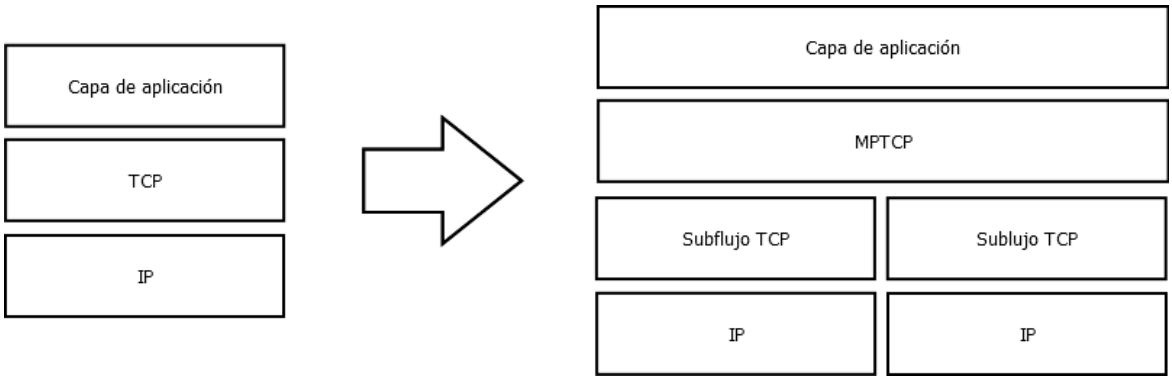



*Figura 3.1: MPTCP en la pila de protocolos*

### 3.2.3 Funcionalidades de MPTCP

A continuación describiremos brevemente las características más importantes del protocolo MPTCP.

1. Establecimiento de la conexión: para respetar la compatibilidad con TCP, una sesión MPTCP se inicia con el tradicional handshake de tres vías. En una segunda fase, si es posible establecer más de un camino, se establecerán otras sesiones TCP.

2. Numeración de la secuencia: MPTCP preserva el esquema de secuenciación de TCP, pero siguiendo uno específico para cada subflujo. Para asegurar la reordenación de los datos en el receptor se requerirá una numeración adicional que permita recomponer las tramas de información que vengan de caminos diferentes. Esta numeración se incluirá dentro de la cabecera MPTCP y se conoce como Data Sequence Numer (DSN).

3. Reordenación de paquetes: el hecho de que la conexión se divida entre varios subflujos sobre diferentes caminos incrementa la posibilidad de recibir segmentos desordenados. Para evitar esto, se definen esquemas que permitan recuperar la información de forma eficiente, para que se minimice la necesidad de retransmisión. El más destacado de estos esquemas es el mecanismo DSACK, una extensión de *Selective Acknowledgement* (SACK) que permite

confirmar y retransmitir selectivamente los fragmentos de información considerados esenciales, evitando la retransmisión de bloques completos.

4. Distribución de paquetes: otro mecanismo necesario, esta vez en el transmisor, es el de distribuir paquetes para ser enviados a través de los subflujos. Persiguiendo una solución simple, se ha optado por un mecanismo *Round Robin*, que distribuye los datos equitativamente entre los caminos disponibles.

5. Además de estas funcionalidades, MPTCP mantiene la mayoría de mecanismos básicos de TCP, como son *slow start, fast retransmit* o *fast recovery*.

Una vez analizado este conjunto de características, parece claro que MPTCP supone una mejora respecto al TCP clásico. Para encontrar los múltiples caminos requeridos por el protocolo se emplean, a nivel de red, los conocidos como Algoritmos Multicamino, que permiten encontrar caminos más o menos disjuntos (es decir, con un menor o mayor número de nodos en común) dentro de una red de topología mallada. Hablaremos de ellos a continuación.

## 3.3 Algoritmos Multicamino

El estudio del protocolo MPTCP nos ha demostrado que podemos encontrar soluciones reales para soportar una sesión de transporte fiable sobre varios caminos simultáneamente. Para ello, sin embargo, necesitamos utilizar protocolos de encaminamiento, diferentes a los clásicos, que encuentren un conjunto de caminos óptimos disjuntos para repartir la carga de tráfico utilizando varios subflujos en un escenario multi-salto. Estos algoritmos se conocen como *link*, *node* y *zone disjoint*.

Tomemos el grafo G(V,E) que representa el escenario en el que queremos obtener un conjunto de caminos empleando dichos algoritmos, entre los nodos fuente (S) y destino (D). V representa el conjunto de vértices (nodos) desplegados en el escenario y E el conjunto de enlaces. Consideraremos que hay un enlace entre dos nodos si están separados por una distancia menor al radio de cobertura, que será el mismo para todos los nodos de la red.

El primer paso será emplear el algoritmo de Dijkstra para encontrar el camino más corto entre S y D. Una vez hecho esto, los algoritmos disjoint eliminarán los enlaces y/o nodos del camino mínimo encontrado (según el caso) y volveremos a aplicar Dijkstra hasta que no tengamos más posibles caminos. Cada proceso se explica con más detalle en los subsiguientes apartados, si bien comenzaremos hablando brevemente del algoritmo de Dijkstra, ya que su ejecución es una parte fundamental del proceso.

**Algoritmo de Dijkstra**

El algoritmo de Dijkstra permite encontrar el camino de coste mínimo desde un nodo origen N hasta todo el resto de nodos de un grafo.

Su funcionamiento se basa en almacenar dos estructuras que incluyen la distancia mínima actual descubierta hacia cada nodo y el camino que hay que seguir para obtener dicha distancia. En cada iteración se busca el nodo con una distancia mínima hasta la fuente, y se exploran todos sus vecinos. Si la distancia para ir al vecino desde la fuente pasado por el nodo es menor a la que había guardada como distancia mínima a dicho vecino, esta se actualiza.

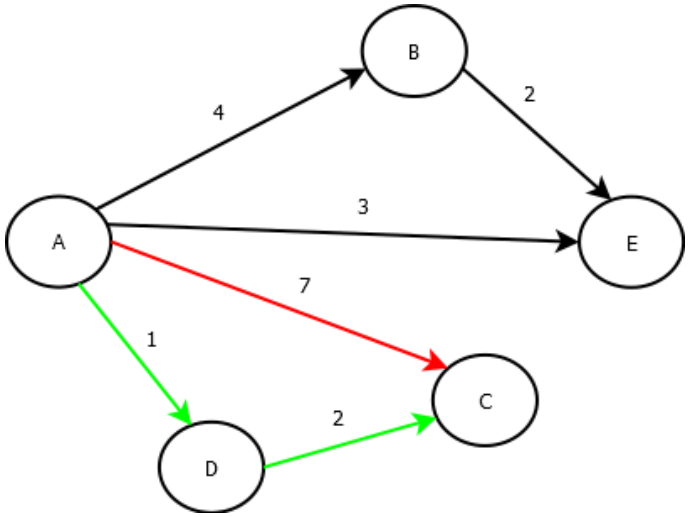



*Figura 3.2: en la iteración 2 del algoritmo de Dijkstra utilizando A como fuente analizamos los vecinos del nodo D. La distancia que teníamos guardada para el nodo C desde A era 7, mayor que la que obtenemos pasando por D que es 2+1=3. Por tanto, actualizamos la distancia a C y especificamos que el camino mínimo a C pasa por D.*

Mediante el algoritmo de Dijkstra las soluciones disjoint buscarán las rutas mínimas entre nodos. Cuando encuentren una ruta mínima eliminarán un conjunto de enlaces para obtener una nueva versión del grafo, y volverán a aplicar Dijkstra hasta hallar un conjunto de caminos disjuntos entre sí. Veamos con más detalle este proceso.

**Algoritmo Link-Disjoint**

El primero de los algoritmos multipath que aplicamos a nuestra red es el conocido como *link disjoint.* Consiste en hallar el conjunto $P_L$ de caminos de coste mínimo entre dos nodos *s* y *d* dentro de un grafo *G(V,E)* dado que sean disjuntos en cuanto a los enlaces que unen cada uno de los nodos intermedios. Es decir, ninguno de los caminos puede compartir enlace con otro, aunque sí pueden tener nodos en común. Su operativa es:

1. Hallar el camino mínimo entre *s* y *d*.
2. Eliminar todos los enlaces del grafo *G(V,E)* original que forman parte del camino hallado en (1) mediante el algoritmo de *Dijkstra*, obteniendo un nuevo grafo $G_L$ (V, $E_L$) sin dichos enlaces.
3. Volver a realizar los pasos (1) y (2) hasta que el algoritmo de *Dijkstra* no encuentre caminos entre *s* y *d*.

De esta forma, $P_L$ estará formado al finalizar el algoritmo por todos los caminos link-disjoint del grafo entre los nodos *s* y *d* (figura 3.3).

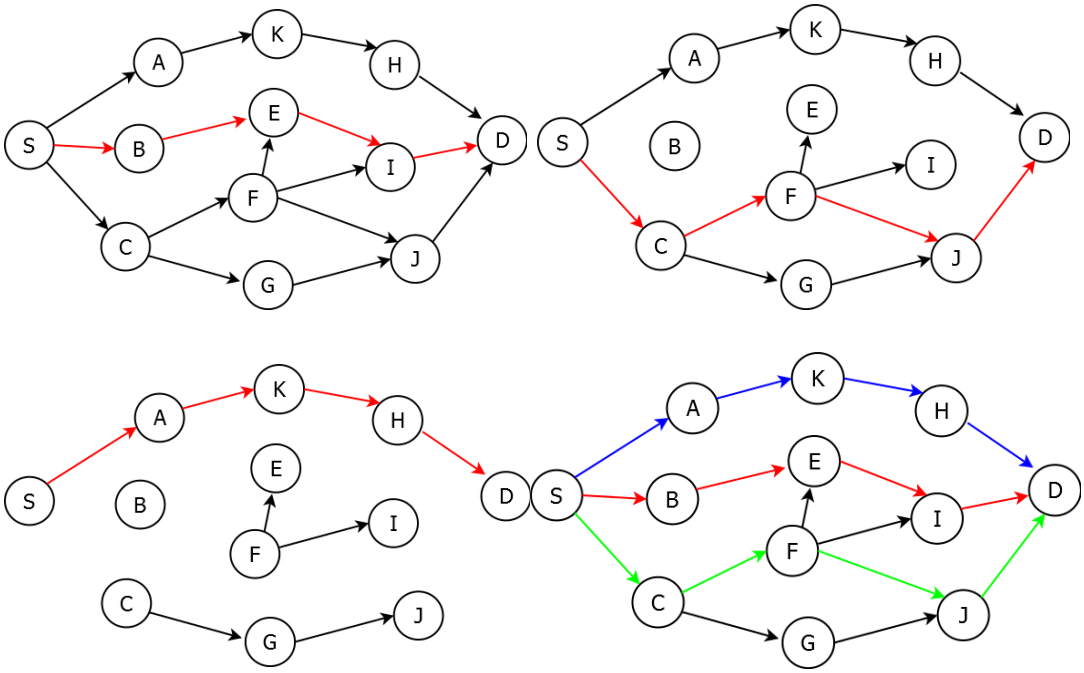



*Figura 3.3: Iteraciones y resultado de aplicar el algoritmo link-disjoint.*

**Algoritmo Node-Disjoint**

El siguiente algoritmo que presentamos es el *node-disjoint*. Mediante este algoritmo obtendremos todos los caminos disjuntos del grafo que no compartan ningún nodo entre sí. Para ello, en cada bucle de ejecución operaremos de forma similar al algoritmo link disjoint, pero en este caso eliminando no sólo los enlaces de cada camino mínimo sino también los nodos entre ellos, exceptuando por supuesto el fuente y el destino. Guardaremos todos los caminos encontrados en una estructura $P_N$.

El proceso será, por tanto:

1. Hallar el camino mínimo entre *s* y *d* mediante el algoritmo de *Dijkstra,* almácenándolo en $P_N$.
2. Eliminar todos los enlaces y los nodos del grafo *G(V,E)* original, exceptuando los nodos extremos *s* y *d*. Se obtiene así un grafo modificado $G_N$ (V, $E_N$) con los enlaces y nodos restantes.
3. Repetir los pasos (1) y (2) hasta que el algoritmo de *Dijkstra* no encuentre caminos entre *s* y *d*.

La representación de la ejecución del algoritmo se observa en la siguiente figura:

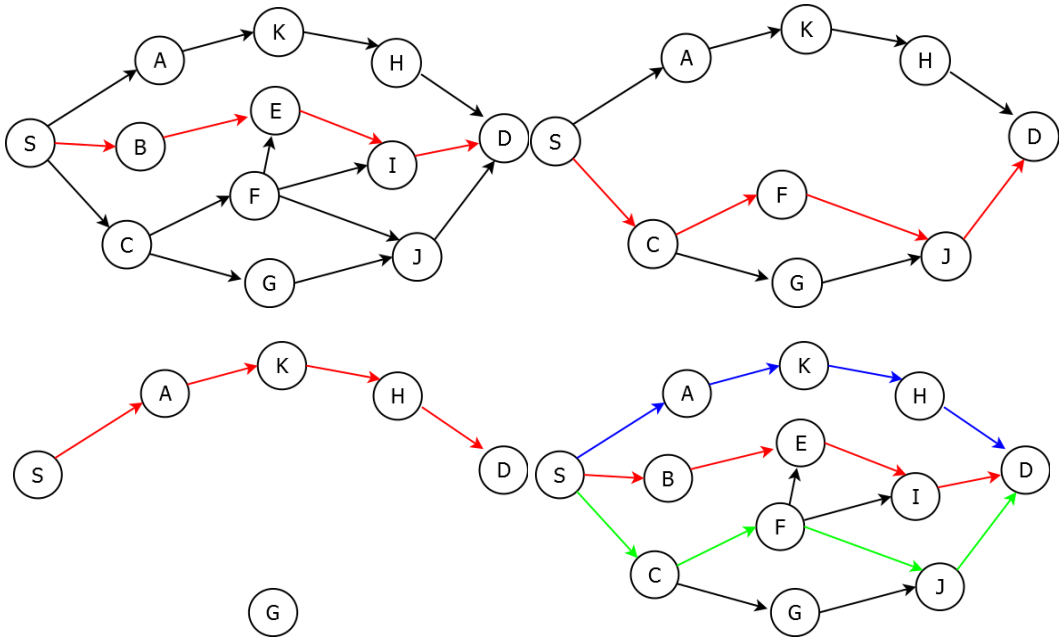



*Figura 3.4: Iteraciones y resultado de aplicar el algoritmo node-disjoint.*

**Algoritmo Zone-Disjoint**

El último (y más restrictivo) de los algoritmos multipath estudiados es el zone-disjoint. Elimina en cada ejecución tanto los enlaces y nodos pertenecientes al camino mínimo encontrado por Dijkstra, como los nodos adyacentes a dicho camino (exceptuando los adyacentes a fuente y destino que no pertenecen al camino).

Su flujo de ejecución es el siguiente:

1. Hallar el camino mínimo entre *s* y *d* mediante el algoritmo de *Dijkstra,* almácenándolo en PZ.

2. Eliminar del grafo:
   a. Los enlaces pertenecientes al camino mínimo.
   b. Los nodos del camino (exceptuando *s* y *d*).
   c. Los nodos adyacentes a los pertenecientes al camino, exceptuando los vecinos de *s* y *d* que no pertenezcan a éste.

   Obtendremos así un grafo modificado $G_Z(V, E_Z)$con los enlaces y nodos restantes.

3. Repetir los pasos (1) y (2) sobre el grafo modificado hasta que *Dijkstra* no encuentre caminos entre *s* y *d*.

La representación gráfica de la ejecución del algoritmo zone-disjoint es la siguiente:

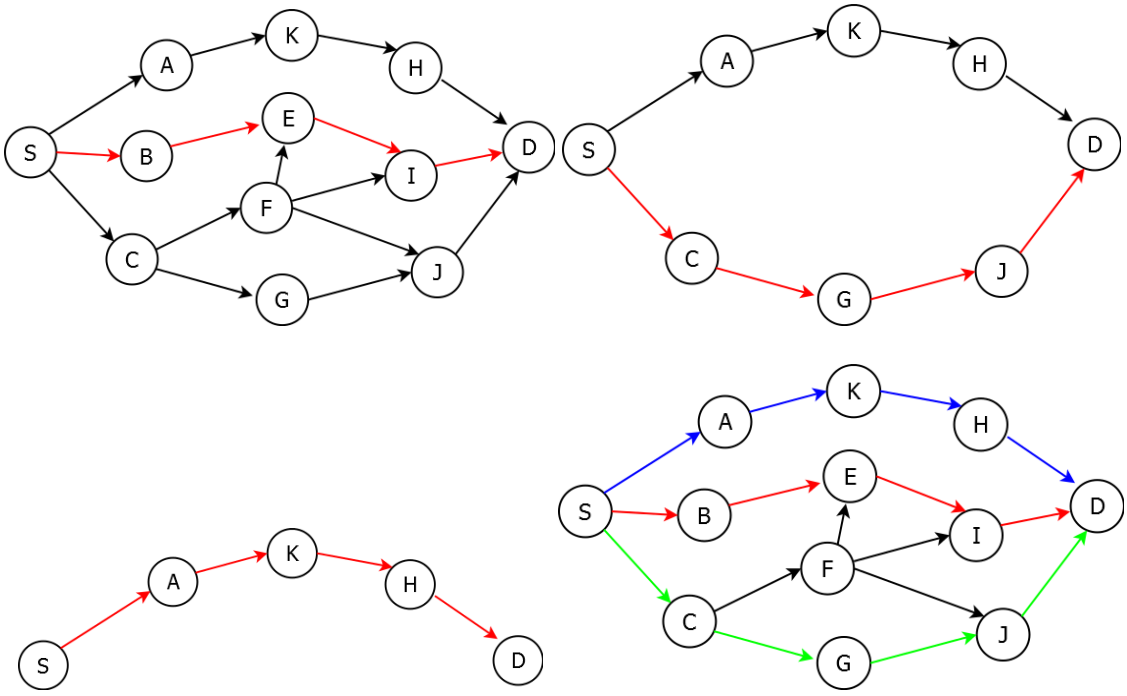



*Figura 3.5: Iteraciones y resultado de aplicar el algoritmo zone-disjoint.*

Analizando el comportamiento de los algoritmos sobre la red de ejemplo podemos anticipar una serie de conclusiones que nos serán de ayuda a la hora de comprobar la utilidad de los algoritmos en redes más complejas:

1. Si bien el proceso de eliminación de enlaces y nodos es diferente para los tres algoritmos, se pueden esperar resultados similares según la topología de la red. En el ejemplo, tanto el algoritmo link-disjoint como el node-disjoint producen el mismo conjunto de caminos como resultado, aunque el node-disjoint descarte los nodos que va utilizando en cada iteración y el link-disjoint no lo haga.

2. El grado de las restricciones de los algoritmos a la hora de descartar caminos es creciente (siendo el link-disjoint el menos restrictivo y el zone-disjoint el más restrictivo). Por tanto, es de esperar que el zone-disjoint nos ofrezca una menor cantidad de posibles paths que los otros dos. Sin embargo, estas rutas esta´rian menos afectados por las interferencias de los otros que en el caso de node-disjoint o link-disjoint.

**Implementación de los algoritmos**

Una vez planteados los tres algoritmos se procedió a su implementación en C++. El objetivo fue crear tres módulos independientes, uno para cada algoritmo, que recurriesen al mismo módulo Dijkstra para elaborar la tabla de rutas.

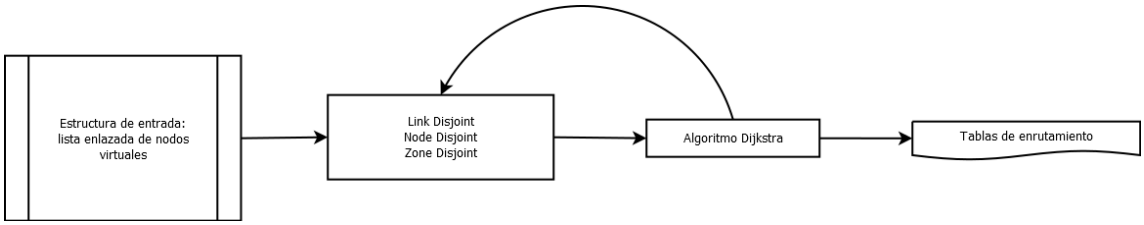



*Figura 3.6: Esquema de funcionamiento de los algoritmos.*

En primer lugar se partió de la estructura de lista enlazada generada en el capítulo anterior: un conjunto de nodos virtuales con sus enlaces y costes.

Los algoritmos Disjoint acceden a dicha estructura y le aplican el mismo algortimo de Dijkstra, procediendo después a eliminar enlaces según hemos comentado en el apartado anterior. Este, cada vez que encuentra una ruta nueva, genera una entrada en el fichero de configuración routing.conf que se enviará posteriormente al simulador. Las entradas en el fichero de configuración se realizaron en base a la red original; esto es, no se indican los nodos virtuales que forman los caminos sino que se realiza la conversión inversa para tratarlos de nuevo como interfaces. De esta forma conseguimos reducir la complejidad de la red que simularemos.

# 4. Simulación de escenarios mediante el simulador NS-3. Análisis de resultados.

Una vez hemos expuesto las características y protocolos de las redes que queremos estudiar, el siguiente paso para comprobar la viabilidad de nuestro estudio es realizar un número de simulaciones estadísticamente representativo, con el que apoyar nuestras suposiciones acerca de las ventajas de usar protocolos multicamino en redes inalámbricas multicanal.

Para ello, se empleará como herramienta el simulador *Network Simulator* en su versión 3. Desarrollado por el Lawrence Berkeley National Laboratory, se trata de un sistema empleado con gran frecuencia dentro de la comunidad científica, principalmente debido a que es una herramienta gratuita, lo cual ha favorecido que su funcionalidad se haya ido ampliando gracias a las contribuciones que se van incorporando a su distribución.

Si bien el papel del simulador NS-3 es crucial en el desarrollo de este TFG, el funcionamiento de dicha herramienta no es objeto de estudio en nuestro caso. Por tanto, el simulador se trató en todo momento como una "caja negra" o sistema cerrado el cual nos proporciona una serie de salidas como respuesta a ciertas entradas, sin que sea necesario conocer al detalle su funcionamiento.

## 4.1 Preparación de las simulaciones. Consideraciones previas.

El primer paso para realizar las simulaciones de los escenarios que a continuación se propondrán fue adecuar las estructuras y ficheros de datos con los que habíamos trabajado hasta ahora a un formato legible por el simulador Network Simulator. De esta forma podemos reflejar nuestro escenario y realizar una gran cantidad de pruebas de forma sencilla. Para ello construimos tres ficheros:

Fichero *scenario.conf*

El objetivo de este fichero es proporcionar al simulador la posición de todos los nodos presentes en el grafo, mediante sus coordenadas euclídeas. Asimismo, incluye cuáles son los nodos cuya comunicación se va a simular, actuando el resto como nodos intermedios en la transmisión. Debido a que nuestros ficheros de entrada en el sistema ya incluían las posiciones en el espacio de los nodos, adaptar el formato fue algo trivial.

Fichero *multipath-channel.conf*

En este caso se quiere informar al simulador de qué nodos poseen un enlace entre ellos. Para ello tenemos que fijarnos tanto en la distancia entre cada par de nodos (tiene que ser menor que el radio de cobertura especificado) como en la compartición de algún canal (esto sólo en el caso de emplear canales diferentes para los nodos, en caso de que todos los nodos incorporen los mismos canales no es necesario comprobarlo).

El fichero representa los enlaces mediante una matriz de adyacencia binaria, en la cual el elemento *(i,j)* es 1 si existe enlace entre los nodos *i* y *j*.

Fichero *routing.conf*

El último fichero fue el que significó una mayor dificultad, puesto que el simulador requería que las rutas incluyesen los nodos originales del grafo real (y no las rutas “virtuales” proporcionadas por nuestros algoritmos de búsqueda de múltiples caminos). Por tanto, hubo que realizar la conversión inversa de nodos virtuales hacia nodos reales e interfaces. Pongamos un ejemplo.

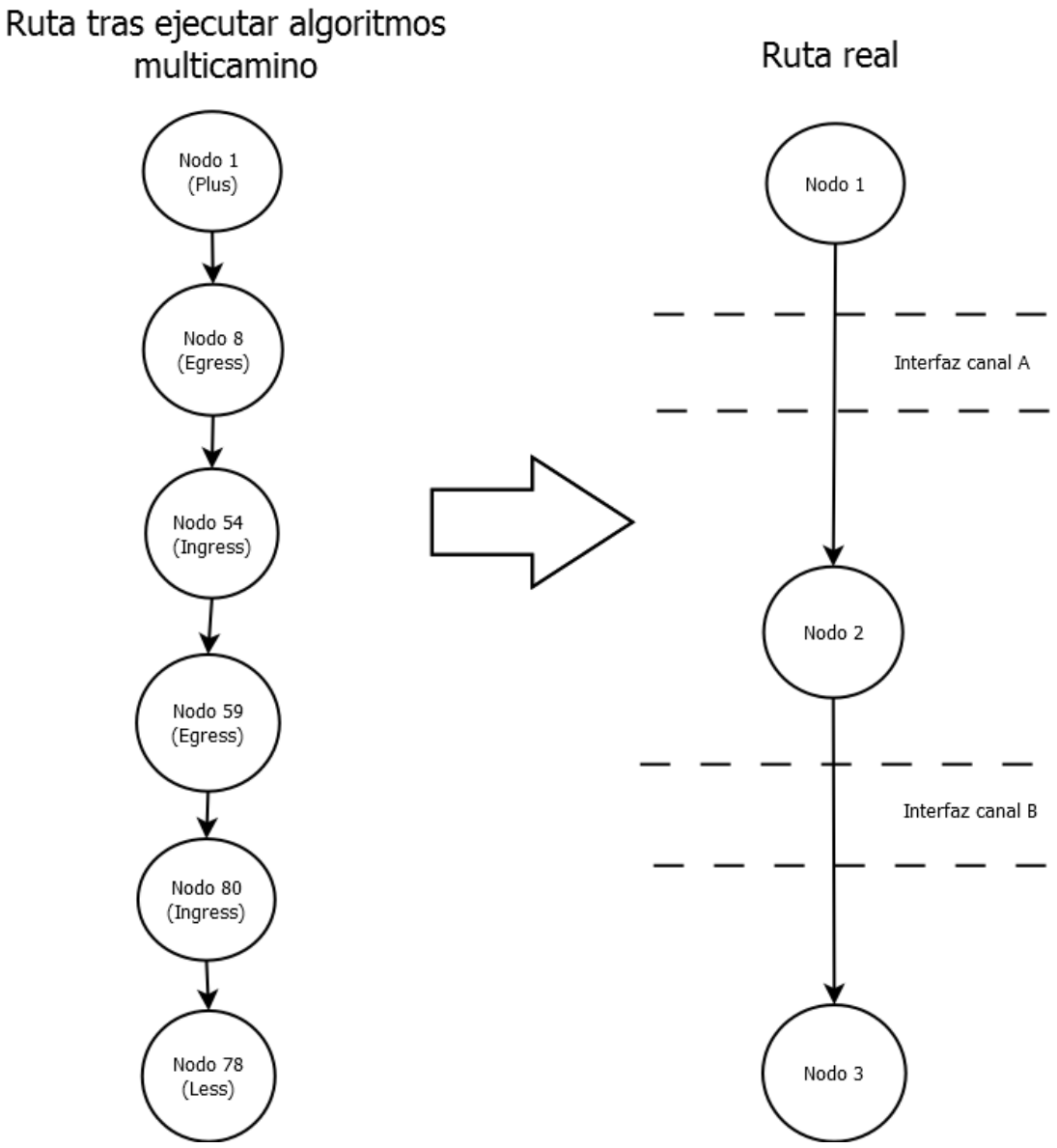



*Figura 4.1: Adecuación del formato de las rutas al simulador NS.*

Una vez tuvimos listos los tres ficheros de configuración, nos encontramos con la siguiente cuestión: ¿qué tipo de escenarios queremos simular? En un principio se consideró la posibilidad de utilizar escenarios prefijados, pero rápidamente se descartaron debido a su escasa relevancia estadística, con lo cual se emplearon únicamente en la fase de desarrollo.

Se decidió, por tanto, la opción de construir escenarios aleatoriamente.

**Disposición de los nodos en el espacio**

A la hora de generar escenarios aleatorios, se plantea en primer lugar la opción más obvia: distribuir todos los nodos de forma completamente aleatoria en un escenario de superficie rectangular. Esta opción, aunque simple, dista mucho de ofrecer datos útiles por dos motivos:

- En primer lugar, si la posición de los nodos origen y destino varía estamos variando las condiciones de análisis en cada escenario.
- En segundo, una distribución puramente aleatoria puede dar lugar a grafos no conectados, es decir, grafos que poseen nodos fuera de alcance del resto. Eso provocaría que la fuente y el destino no puedan comunicarse, dejando el escenario inservible.

Por tanto, una distribución puramente aleatoria parece no ser la más adecuada. Para evitar los dos problemas mencionados optamos por:

a. Fijar la posición de los nodos origen y destino. De esta forma la distancia a cubrir entre ambos nodos es siempre la misma y reflejamos un posible escenario real en la comunicación entre dos nodos fijos a través de una red mallada.

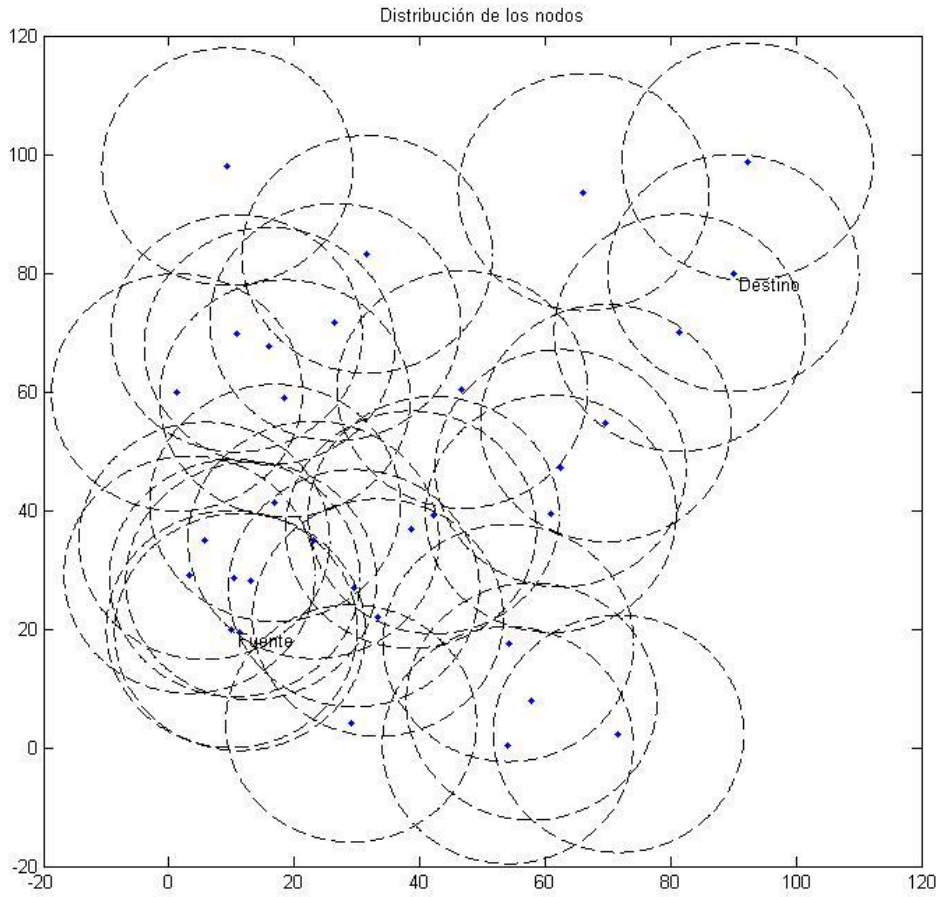



*Figura 4.2: Distribución de 32 nodos en un escenario de 100 x 100 metros con 20 metros de radio de cobertura de forma aleatoria, exceptuando los nodos fuente (10,20) y destino (80,90).*

b. Asegurar que el grafo es completamente conexo. Para ello se modificó la función de generación aleatoria que mencionamos en el capítulo 2. El proceso de generación de un grafo que se utilizó fue el siguiente:

   1. Situar los nodos fuente y destino en las posiciones preestablecidas.

   2. Ir añadiendo nodos al grafo de uno en uno, comprobando que están conectados a este, es decir, midiendo la distancia a los nodos ya generados y asegurándose de que esta es menor que el radio de cobertura.

   3. Una vez añadidos todos los nodos que queremos, comprobar que existe una ruta desde la fuente al destino. Esto es necesario debido a que añadiendo los nodos uno a uno corremos el riesgo de que se forme una "nube" de nodos en torno a fuente o destino, quedando el otro aislado.

   4. En caso de que no haya ruta, repetir el proceso.

De esta forma obtenemos grafos completamente conexos y con una ruta garantizada entre los nodos que se comunican:

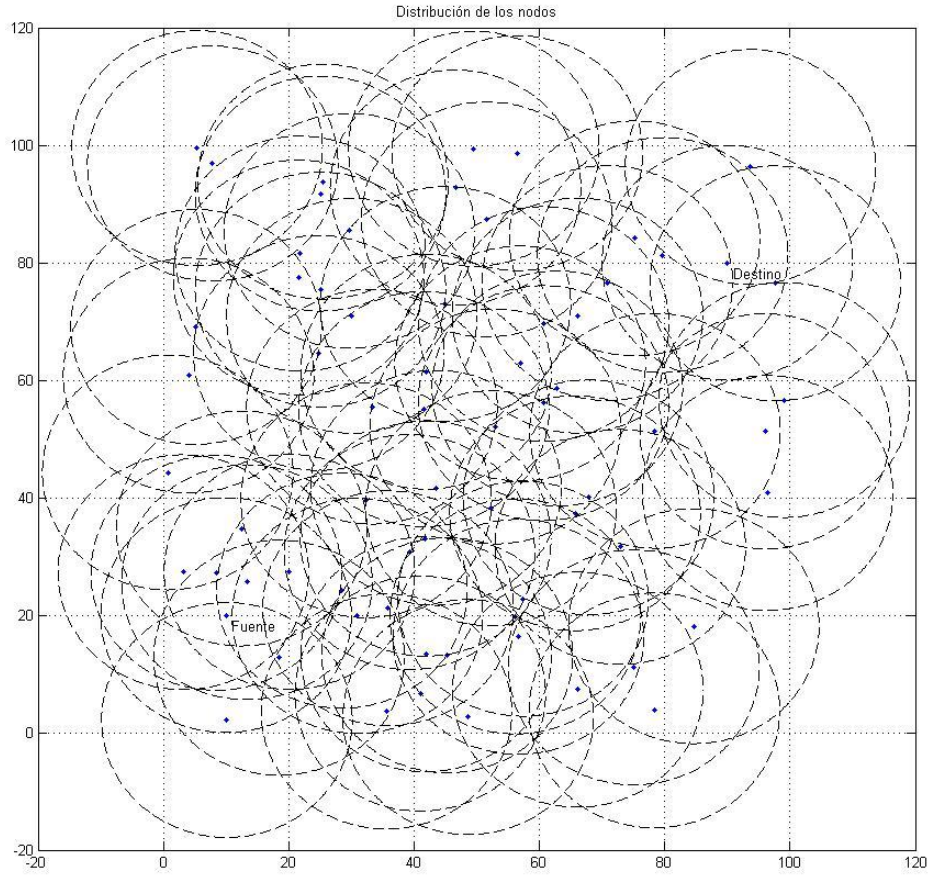



*Figura 4.3: Distribución que garantiza que el grafo es completamente conexo y existe una ruta entre los nodos fuente (10,20) y destino (90,80).*

**Selección de los canales**

En anteriores trabajos realizados dentro del Grupo de Ingeniería Telemática la generación del escenario se había limitado al paso anterior, es decir, a encontrar una distribución espacial apropiada. Sin embargo, en este Trabajo de Fin de Grado que tiene por objetivo probar el rendimiento de los algoritmos multicamino en redes multicanal, la elección de los canales no es algo arbitrario. Tenemos dos posibilidades a la hora de definir nuestros escenarios:

- Que todos los nodos de la red empleen los mismos canales. En ese caso, estaríamos suponiendo que todos los nodos son iguales y poseen varias interfaces que pueden comunicarse entre sí. Un ejemplo sería una red mallada en la cual cada dispositivo posee varias interfaces radio configurada para funcionar en uno de los canales independientes de 802.11. Todos los nodos pueden comunicarse entre sí siempre y cuando estén dentro de la distancia de cobertura.

- Que, dentro de un conjunto de interfaces predefinidas, cada nodo pueda incorporar o no (aleatoriamente) cada una de ellas. Ese caso incluye una dificultad adicional a las comunicaciones: aunque dos nodos se encuentren a una distancia menor al radio de cobertura, puede que no compartan ninguna de las interfaces y, por ende, no puedan comunicarse entre sí. Este tipo de despliegue resulta de gran interés dentro del modelo con el que estamos trabajando, ya que plantea la posibilidad de comunicar dispositivos utilizando diferentes tecnologías simultáneamente.

Se acordó realizar simulaciones de los dos escenarios posibles, si bien el número de escenarios fue mayor para el segundo caso debido a que desde un principio suscitaba mayor interés.

## 4.2 Simulaciones realizadas

Una vez elegidos todos los parámetros con los que vamos a configurar nuestros escenarios, el siguiente paso fue introducir todos los datos tanto en nuestra implementación en C++, como en el simulador NS-3, para obtener los resultados correspondientes.

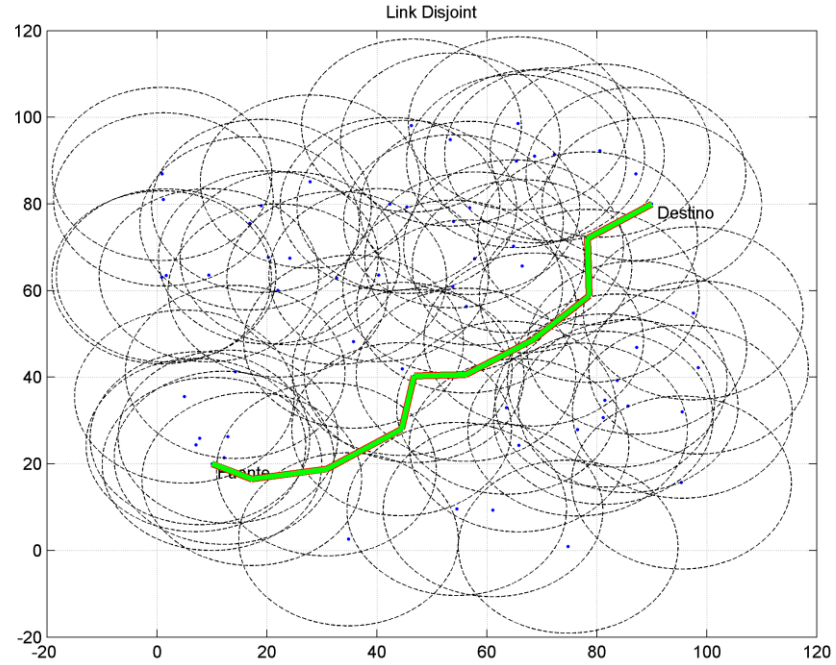
### 4.2.1 Redes con canales uniformes

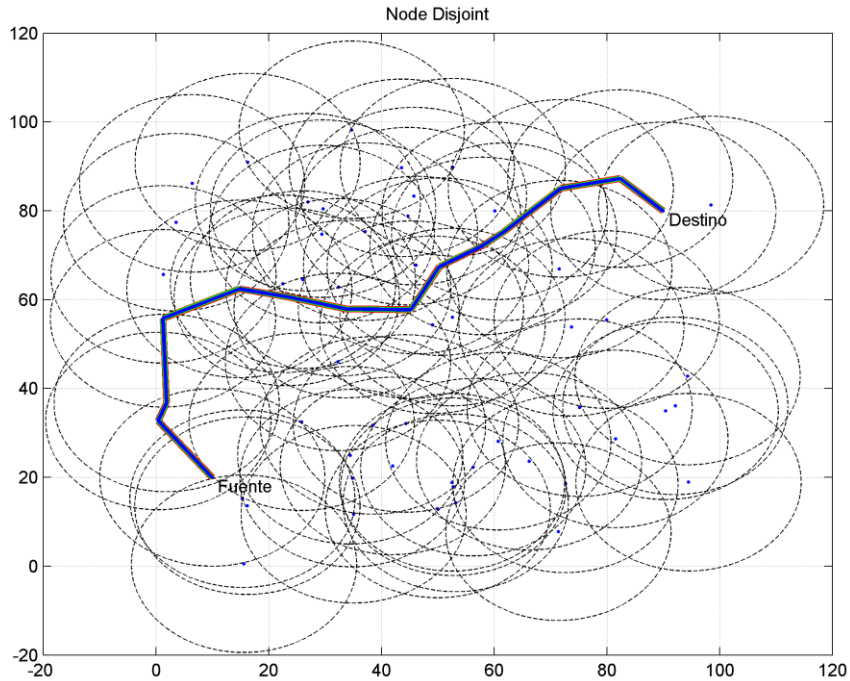
Las primeras simulaciones que se efectuaron fueron creando una red de 64 nodos en la cual todos ellos son capaces de comunicarse por los mismos canales, para el caso de 2 y 3 canales. Para cada uno de esos dos casos, se realizaron 50 simulaciones con cada uno de los tres algoritmos disjoint y se estudió su comportamiento.

**Algoritmos Link-Disjoint y Node-Disjoint**
Agrupamos la respuesta de los dos primeros algoritmos, puesto que su funcionamiento con este tipo de canales resultó ser muy similar. Ambos protocolos son capaces de encontrar el mismo número de rutas en todos los escenarios, que no es otro que el número de canales disponibles en cada caso.

$$Nrutas_{L-D} = Nrutas_{N-D} = Ncanales \tag{4.1}$$

Si echamos un vistazo a las rutas generadas, veremos que en la gran mayoría de los casos ambas rutas pasan por los mismos nodos, que son los encontrados por el algoritmo de Dijkstra como camino mínimo dentro del grafo. Veamos a continuación dos ejemplos:




*Figuras 4.4 y 4.5: Rutas encontradas por LD y ND en redes con canales uniformes (2 y 3 canales, respectivamente). Primera ruta en rojo, segunda en verde y tercera en azul.*

Como podemos observar, en los dos casos las rutas que obtienen los algoritmos Link y Node Disjoint son iguales al número de canales y atraviesan los mismos nodos. Esto es debido a que el coste introducido por emplear una ruta de mayor distancia es superior al de cambiar de canal dentro de la misma ruta (recordamos el parámetro alfa dentro del apartado de balanceo de carga en el capítulo 2).

Si observamos las rutas generadas en uno de los escenarios, podemos ver que siguen el siguiente patrón: se trata de evitar en la medida de lo posible la repetición de canales en saltos consecutivos (y si es posible también la repetición del canal predecesor). Por tanto se demuestra que la métrica CSC se tiene en cuenta y se reduce la interferencia intra-flow.

En ciertos casos, aproximadamente en el 10% de las simulaciones, se obtuvieron puntos en los cuales emplear una ruta diferente suponía una mejora en el rendimiento. Veamos a continuación un ejemplo.

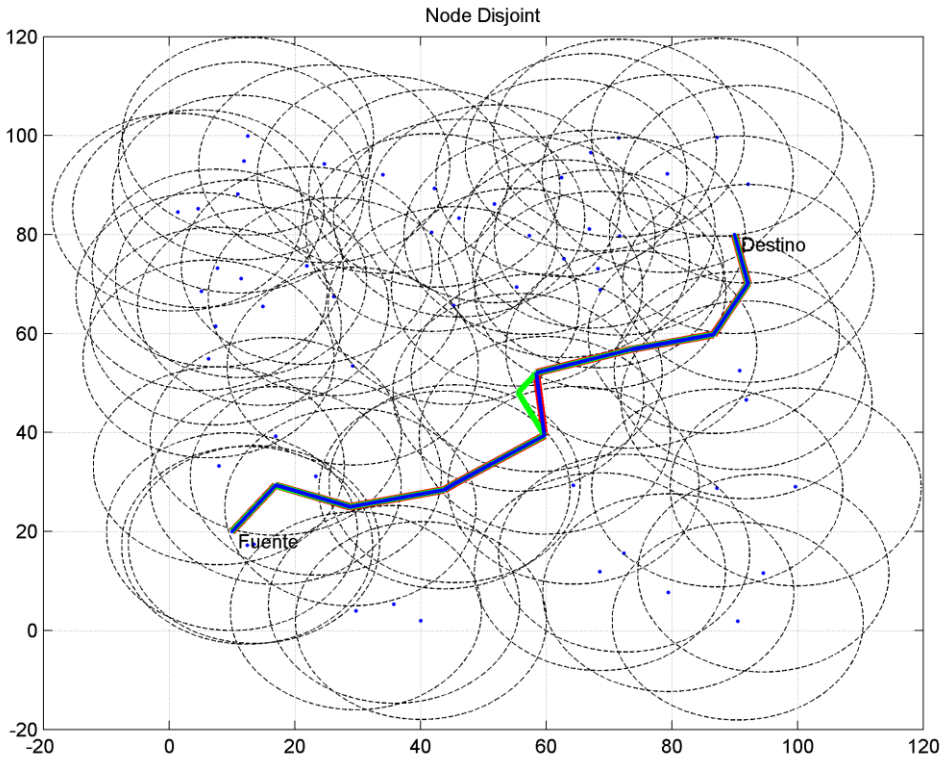
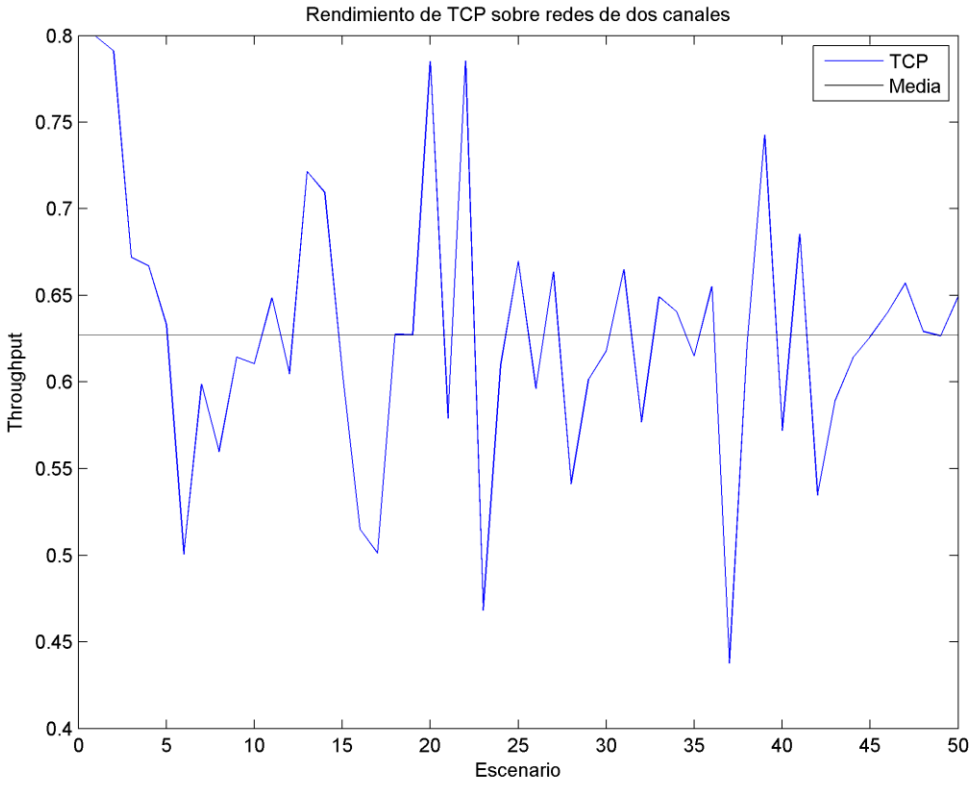
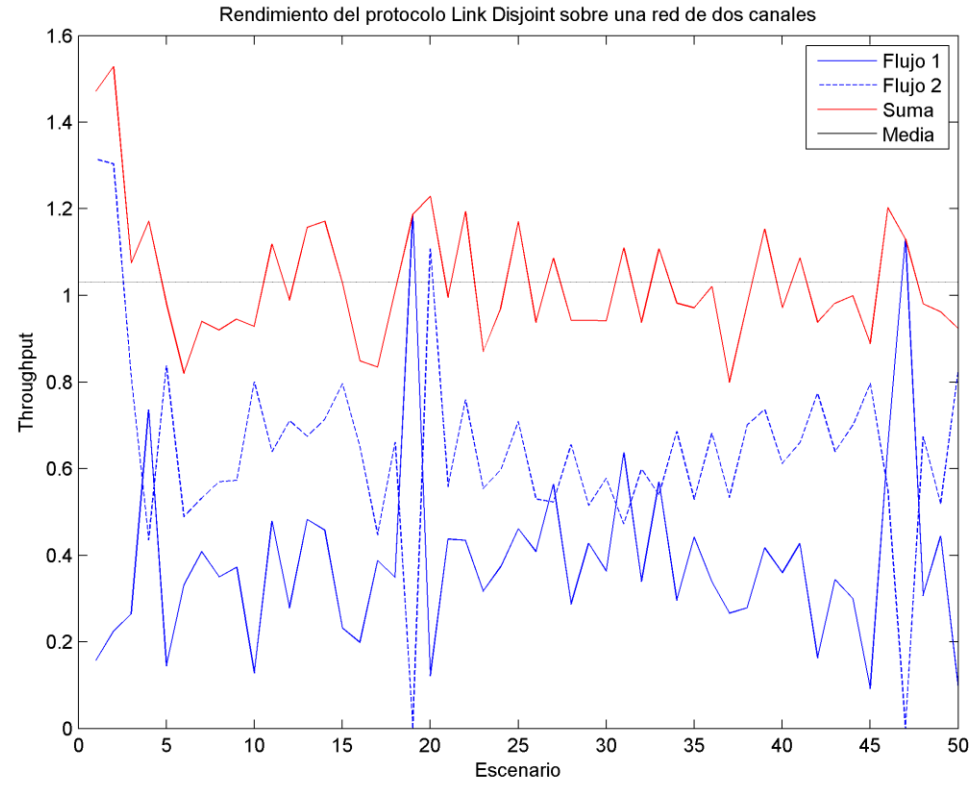



*Figura 4.6: podemos observar cómo en la segunda ruta encontrada por el algoritmo Node Disjoint difiere de la principal y se añade un salto adicional.*

Una vez introducidos este conjunto de escenarios en el simulador obtuvimos una importante mejora en el rendimiento respecto a TCP. En las figuras 4.7 y 4.8 vemos por un lado el rendimiento de TCP y por otro el de Link Disjoint sobre una red de dos canales. Como se puede apreciar, la media del rendimiento en 50 escenarios es casi el doble en el caso del protocolo multicamino.




*Figuras 4.7 y 4.8: rendimiento de TCP clásico y Link Disjoint en una red de dos canales homogénea.*

**Algoritmo Zone Disjoint**

El algoritmo Zone Disjoint tiene un comportamiento muy diferente a los otros dos cuando se trata de redes con un número de canales homogéneo en los nodos. Para empezar, si ejecutamos dicho algoritmo sobre las redes de 2 canales, vemos que en ningún caso se pueden obtener múltiples rutas. Esto se debe a que, al eliminar los vecinos de los nodos ingress en la topología y poseer solo dos canales, se agotan las vías de entrada a todos los nodos. Por tanto, parece claro que la utilización de este algoritmo no es útil en ese caso concreto.

Sin embargo, al emplear 3 canales nos encontramos con un caso completamente distinto: el algoritmo Zone Disjoint es capaz de encontrar múltiples caminos en el 100% de las distribuciones de nodos planteadas. Además, el número de rutas coincide con el número de canales en el 80% de los casos. Sin embargo, analizando uno de dichos casos nos encontramos con lo siguiente:

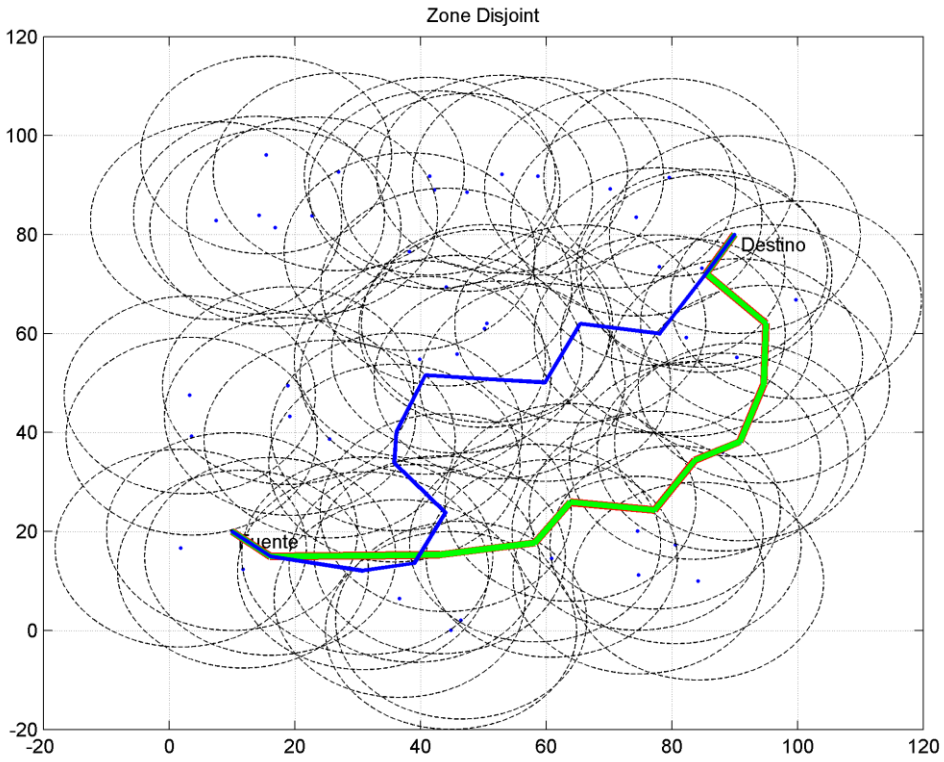



*Figura 4.9: Ejecución de Zone Disjoint en una red homogénea de 3 canales.*

Los canales empleados en cada ruta son:

Ruta 1 (roja): C → B → A → C → B → A → C → B → A
Ruta 2 (verde): A → A → B → C → A → B → C → A → B
Ruta 3 (azul): canal B en todos los saltos.

Como podemos apreciar, se obtienen dos rutas en las que se emplean interfaces alternas y una tercera que utiliza siempre la misma interfaz, pero siguiendo un conjunto

de nodos diferente. En esa tercera ruta la interferencia *intra-flow* será un factor a tener en cuenta, puesto que todos los nodos emplearán el mismo canal. Esto podría dar lugar a un aumento considerable en la tasa de errores de la comunicación. Sin embargo, el hecho de usar nodos diferentes paliaría en gran medida las interferencias con los otros caminos, lo cual propone un escenario interesante.

**Valoración general de los protocolos**

Una vez tuvimos listos los escenarios y rutas que queríamos simular, el siguiente paso fue introducirlos en el Network Simulator 3 para comprobar qué rendimiento real ofrecen nuestros algoritmos disjoint. Los parámetros de configuración del simulador fueron los mismos para todos los resultados que aparecen a continuación:

Tamaño del paquete 1460 bytes (1500 bytes Wi Fi + 40 de la cabecera TCP/IP)
Versión del protocolo 802.11: b (11 Mbps)
Nota: todos los flujos están expresados en Megabits/segundo (Mbps).

Una vez realizamos todas las simulaciones tanto para dos como para tres canales, obtuvimos unos resultados más que prometedores para todos los protocolos:

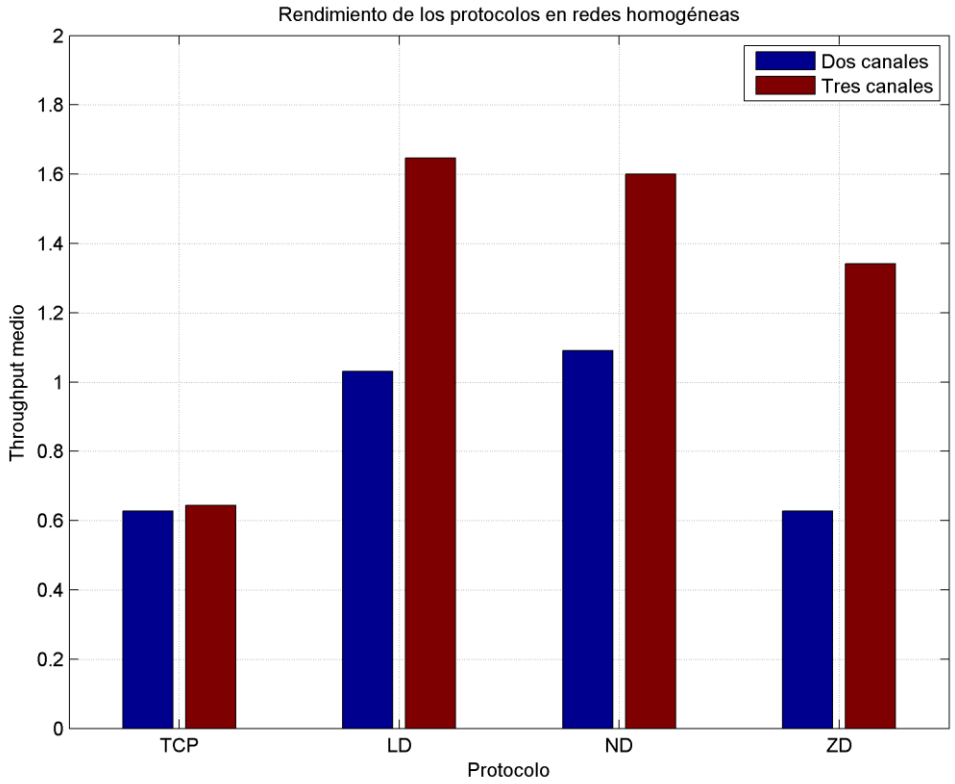



*Figura 4.10: Rendimiento (Mbps) de TCP frente a los protocolos Disjoint en redes homogéneas de dos y tres canales.*

Para poder apreciar con mayor claridad la ganancia en rendimiento obtenida con los protocolos, representamos en la figura 4.11 la ganancia relativa con respecto a aplicar TCP en cada uno de los conjuntos de escenarios.

$$G.R.(\%) = \frac{Throughput(Disjoint) - Throughput(TCP)}{Throughput(TCP)} * 100 \quad (4.2)$$

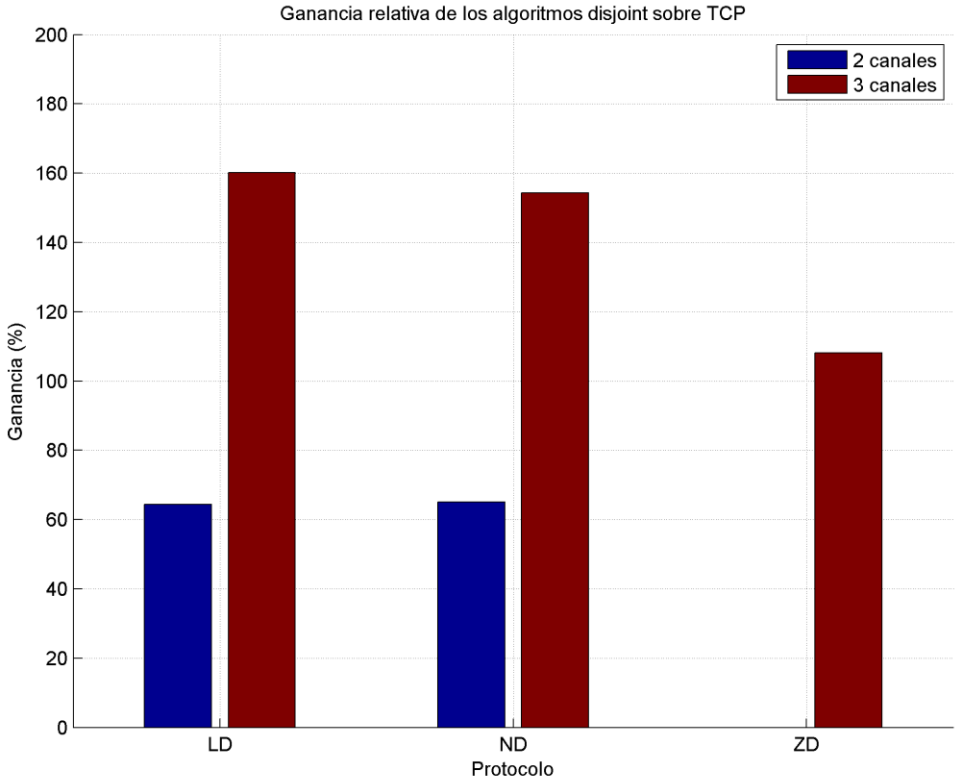



*Figura 4.11: Ganancia relativa de los protocolos disjoint sobre TCP para redes homogéneas.*

Como podemos observar, todos los protocolos, a excepción de Zone Disjoint sobre una red de dos canales, ofrecen una mejora en el rendimiento más que notable; mediante el uso de los algoritmos Node y Link Disjoint podemos llegar a casi triplicar el throughput que obtenemos sobre una red de tres canales de comunicación, lo cual podemos evaluar como unos resultados óptimos.

### 4.2 Redes con canales no uniformes

Nos referiremos así en este apartado a un conjunto de redes diferentes a las anteriores. En este caso, no todos los nodos utilizan todos los canales, sino que incorporan cada uno de ellos según una cierta probabilidad. De esta forma, representamos un posible escenario en el cual cada nodo tiene unas interfaces o tecnologías diferentes, como podría ser una red móvil o una red local en la que ciertos terminales poseen tarjeta inalámbrica y otros no.

¿Cómo se comportan los algoritmos Disjoint en estos casos? Veamos un ejemplo.

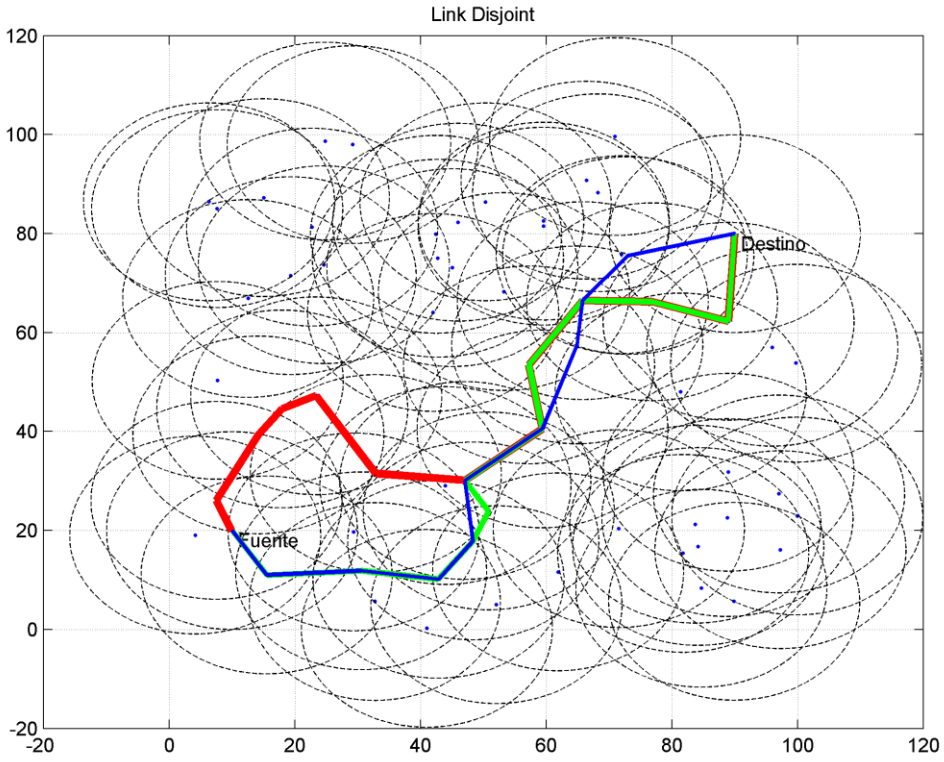



*Figura 4.12: Ejecución de Link Disjoint sobre 64 nodos con canales aleatorios, con una probabilidad del 80% de incorporar cada canal individual.*

Hemos tomado un ejemplo con tres rutas para apreciar claramente el resultado. El algoritmo Link Disjoint es capaz de encontrar tres rutas independientes a través de la red. Sin embargo, eso no es lo habitual puesto que se depende en gran medida de que los nodos incorporen los canales necesarios. En la figura 4.13 podemos ver el diferente rendimiento de los protocolos sobre esta clase de redes, observando el número de rutas medio que obtiene un algoritmo sobre un conjunto de escenarios aleatorios.

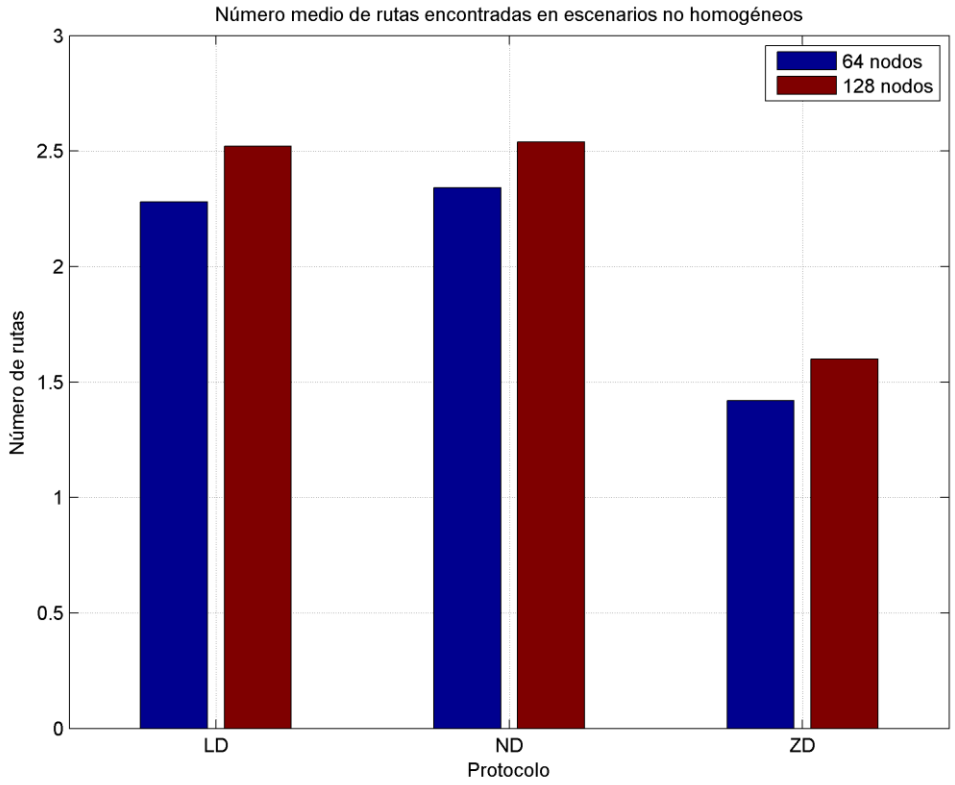
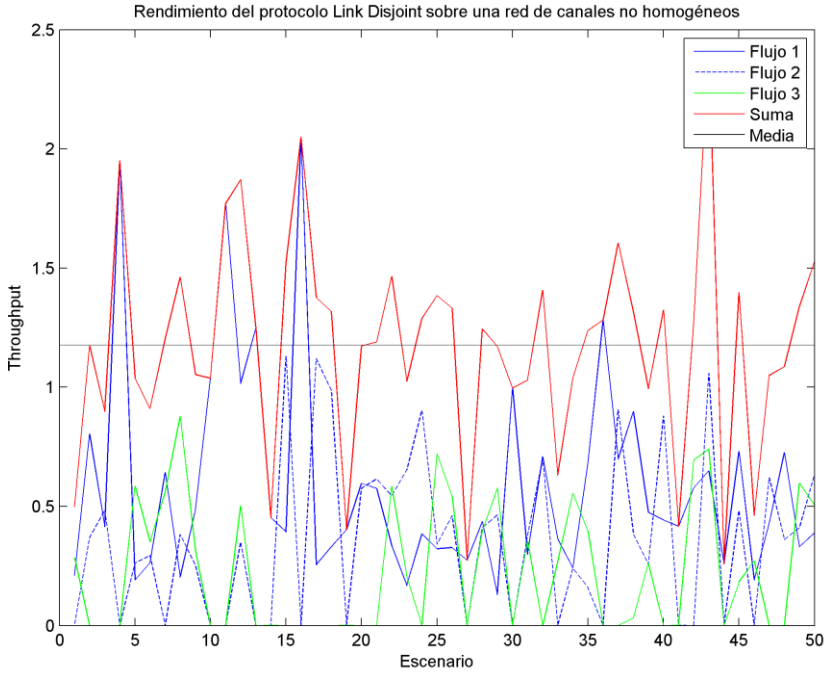


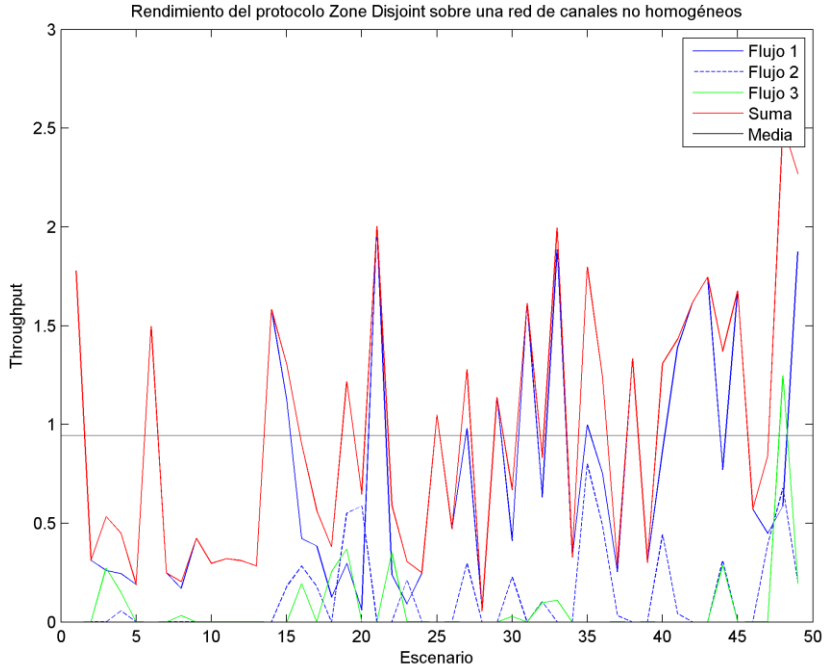

*Figura 4.13: Número medio de rutas encontradas por los protocolos Disjoint en escenarios con canales aleatorios.*

Como podemos apreciar, a medida que introducimos más nodos en el mapa es más fácil que los algoritmos encuentren rutas. Es destacable el rendimiento muy inferior del protocolo Zone Disjoint frente a los otros dos.

**Rendimiento de los algoritmos Disjoint en redes no homogéneas**

Una característica clave de esta clase de escenarios es que el rendimiento de los protocolos está muy ligado a los canales que incorporan los nodos. Al tener cada nodo una probabilidad de 0.8 de utilizar un canal, el rendimiento que obtiene el mismo protocolo en escenarios diferentes puede variar mucho. Veamos dos ejemplos:




*Figuras 4.14 y 4.15: protocolos Link y Zone disjoint sobre redes con canales no uniformes.*

En el eje x podemos ver el conjunto de 50 escenarios que simulamos en cada caso. Parece evidente que las características de cada uno de ellos influyen mucho en el rendimiento, puesto que la suma total de rendimiento de los flujos varía mucho. Esto es especialmente remarcable en el caso de Zone Disjoint, puesto que en la mayoría de los escenarios no existe siquiera una tercera ruta.

En este caso se hace aún más valioso el uso del simulador NS-3. Al realizar pruebas sobre 50 escenarios diferentes (y, sobre cada uno de ellos, 10 simulaciones) nos aseguramos de obtener unos resultados estadísticos relevantes.

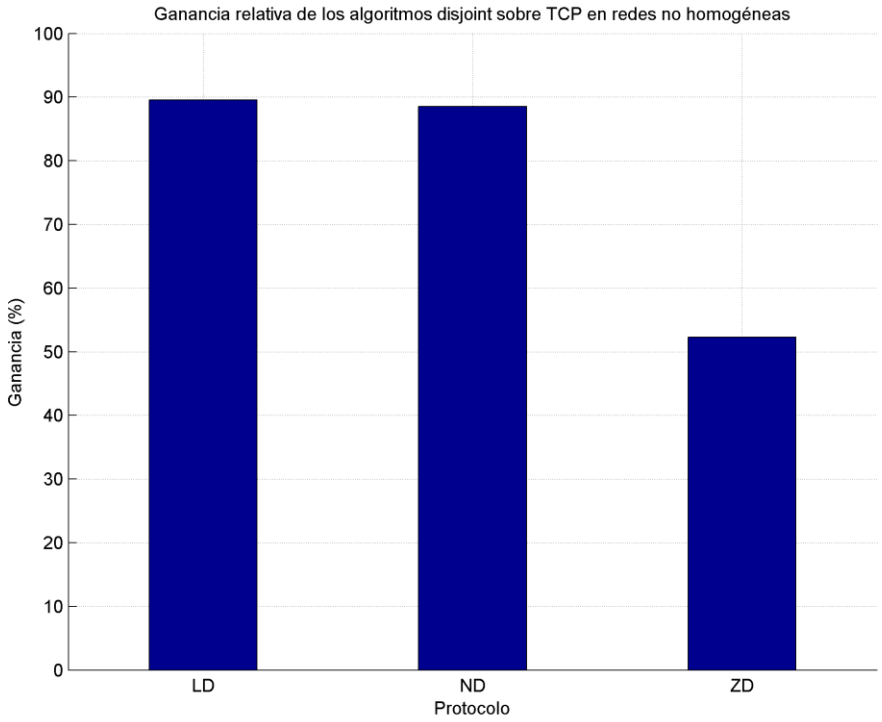



*Figura 4.16: ganancia relativa de los algoritmos disjoint sobre TCP en redes de 128 nodos que incorporan canales de forma aleatoria.*

En este caso la ganancia relativa que obtenemos es menor que en las redes en las que todos los nodos incorporaban los tres canales, lo cual era en cierto modo esperable. Sin embargo, los algoritmos disjoint siguen poniendo de manifiesto un gran comportamiento.

A modo de resumen de este cuarto capítulo, podemos valorar los resultados de las simulaciones como muy positivas. Los algoritmos Disjoint han demostrado ser una alternativa válida al modelo monocanal que empleamos actualmente en la mayoría de redes. Dentro de ellos, la utilización de uno u otro dependerá en gran medida de nuestros objetivos. Si bien Link y Node Disjoint proporcionan unas tasas de transferencia de datos superiores, la mayor robustez de Zone Disjoint (puesto que crea rutas con menos interferencia entre ellas) es un factor a tener en cuenta para casos en los cuales la velocidad de transmisión no sea un factor crítico (lo cual es algo habitual en las redes inalámbricas malladas).

# 5. Conclusiones y líneas de investigación futuras

El desarrollo de este trabajo de fin de grado nos ha servido para aproximarnos un poco más al comportamiento de una tecnología actualmente en auge, la de las redes malladas inalámbricas de múltiples interfaces.

En primer lugar nos apoyamos en el protocolo de balanceo de carga LIBRA y la métrica asociada a éste, MIC, para asegurarnos de construir un conjunto de escenarios que reflejasen con la mayor fidelidad posible el comportamiento de estas redes. Tuvimos en cuenta las limitaciones del canal inalámbrico y el criterio de isotonicidad.

A continuación nos centramos en los algoritmos multicamino, los cuales adaptamos para que funcionasen sobre nuestras nuevas redes. Estudiamos cada uno en particular y realizamos su implementación en lenguaje *C++*.

Por último, realizamos un conjunto significativo de simulaciones en el Network Simulator 3 con el fin de obtener un conjunto de resultados que probasen el valor de emplear múltiples rutas. Los resultados fueron satisfactorios, puesto que los protocolos obtuvieron un rendimiento superior a TCP en la gran mayoría de los escenarios planteados, tanto utilizando redes con canales homogéneos como no homogéneos.

En conjunto, parece claro que el aprovechamiento de múltiples interfaces es el camino a seguir en cuanto a redes inalámbricas se refiere. La mejora de rendimiento en los casos estudiados nos lleva a pensar que el modelo de un sólo canal que gobierna actualmente las comunicaciones presenta claras limitaciones.

En cuanto al siguiente paso, se abren varios caminos a seguir. En primer lugar, las simulaciones realizadas en este trabajo fueron sobre los diferentes canales que nos proporciona una red Wi Fi. Un paso lógico parece aprovechar las diferentes interfaces que incorporan los dispositivos actuales (véanse móviles, ordenadores personales) y las características de transparencia hacia otras capas de MPTCP para repartir tráfico entre diferentes protocolos y tecnologías.

Por otro lado, incorporar el estudio de tasas de errores aportaría una mejor visión a la hora de elegir entre el protocolo Zone Disjoint y el resto, puesto que su mayor robustez no queda reflejada en las redes ideales que manejamos en nuestras simulaciones (siempre tomamos enlaces sin pérdidas).

Asimismo, se podría valorar la presencia de nodos móviles en la red, lo cual es un hecho más que habitual en las redes malladas. De esta forma, habría que introducir el factor de actualización de las rutas y llegar a un compromiso entre rendimiento y generación de tráfico de señalización.

# 6. Bibliografía y referencias

[1] Yaling Yang, Jun Wang, Robin Kravets, “Interference-aware Load Balancing for Multihop Wireless Networks”. University of Illinois, 2005.

[2] A.Ford, C.Raiciu, M. Handley, O.Bonaventure, “TCP Extensions for Multipath Operation with Multiple Addresses”. RFC 6824 of IETF, 2013.

[3] David Gómez, Pablo Garrido, Carlos Rabadán, Ramón Agüero, Luis Muñoz, “TCP Performance Enhancement over Wireless Mesh Networks by means of the Combination of Multi-RAT Devices and the MPTCP Protocol”, Network Protocols and Algorithms, 2014 Vol.6 No. 3.

[4] Damon Wischnik, Costin Raiciu, Adam Greenhalgh, Mark Handley, “Design, implementation and evaluating of congestion control for multipath TCP”, University College London, 2011.

[5] PFC Carlos Rabadán, “Algoritmos para el estudio de técnicas de multi-path y network-coding en redes inalámbricas multi-salto”, Universidad de Cantabria, 2013.

[6] PFC Javier Quintana, “Balanceo de carga en redes inalámbricas multi-salto”, Universidad de Cantabria, 2010.

[7] Tesis Ramón Agüero, “Contribución a la mejora de las prestaciones en redes de acceso inalámbricas no convencionales”, Universidad de Cantabria, 2007.

[8] James F.Kurose, Keith W.Ross, “Redes de Computadoras, un enfoque descendente”, Pearson Educación, 2010.