Facultad de Ciencias

Algoritmo para determinar la posibilidad de aplicar técnicas de codificación de red inter-flujo sobre redes malladas inalámbricas

(Algorithm to assess the feasibility of inter-flow network coding techniques over wireless mesh nerworks)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN MATEMÁTICAS Y COMPUTACIÓN

Autor: Pablo Garrido Ortiz

Director\es: Ramón Agüero Calvo,
Francisco Santos Leal

Julio - 2014

Índice

Índice de figuras

Resumen ejecutivo

Este trabajo pretende estudiar la aplicabilidad de las técnicas de Network Coding sobre escenarios con un carácter aleatorio. Esta técnica, propuestas por Ahlswede, pretende mejorar el rendimiento mediante la combinación de paquetes pertenecientes a diferentes flujos de información. En concreto, se plantea su uso sobre redes malladas inalámbricas.

La demanda de capacidad de los dispositivos inalámbricos ha sufrido un vertiginoso crecimiento en los últimos años y se espera que esta dinámica se mantenga, debido principalmente debe al aumento de dispositivos de comunicaciones inalámbricas y las capacidades que se ofrecen a los usuarios (videos, juegos, redes sociales...). Se marcan dos grandes retos para satisfacer las exigencias del mercado: ampliar la cobertura y mejorar el rendimiento, entendiendo que este último factor es una de las grandes debilidades de las comunicaciones inalámbricas. En el intento de mejorar ambos aspectos surgen las técnicas de Network Coding sobre redes malladas. Por un lado, las redes malladas pretenden aumentar el área de cobertura de despliegues más tradicionales y, por otro, las técnicas de Network Coding buscan mejorar el rendimiento que se alcanza en la comunicación bajo los estándares actuales de IEEE 802.11 (a nivel de enlace) y TCP/IP (a nivel de transporte y red).

Para evaluar la aplicabilidad de estas técnicas se despliegan escenarios aleatorios y se desarrolla un algoritmo que permite evaluar si existe la opción de utilizar Network Coding. A pesar de que los escenarios aleatorios generados tienen ciertas características que favorecen el uso de Network Coding, los resultados no son muy esperanzadores. Se obtiene únicamente un 8 % de escenarios proclives a usar estas técnicas. Además, algunos de los escenarios desplegados se han llevado al simulador ns-3, para analizar qué mejora de rendimiento se podría alcanzar.

Abstract

In this work we asses the feasibility of network coding over random scenarios. This technique, which was proposed by Ahlswede, aim to improve the performance by combining packets, that belong to different flows. In particular, we focus on the use of Network Coding technique over wireless mesh networks.

The capacity required by the wireless devices has remarkably grown during the latest years and such growth is expected to be maintained in the future. This higher demand is a consequence of the increase in the number of devices with wireless communication capabilities and the capacity they offer to the end users (videos, games, social networks, ...). There are two mains issues to satisfy the market requirements: to broaden the network coverage and to enhance the performance. The use of Network Coding techniques over mesh wireless networks looms so as to tackle both goals. On the one hand, wireless mesh network broaden coverage network deployments and, on the other hand, network coding improves the performance offered by current technologies, such as IEEE 802.11 and TCP.

In order to evaluate the feasibility of network coding we deploy random scenarios and we propose an algorithms to establish the possibility of applying network coding. Despite the advantageous features of the deployed random topologies, which benefit the applicability of network coding, the results show that the probability of promoting these techniques is rather low. Moreover, we use the ns-3 simulator to analyze the performance that can be achieved.

Agradecimientos

Primero dar las gracias a los profesores que han sabido transmitir sus conocimientos. En especial a Ramón, Paco y David que me han ayudado en todo momento para sacar el trabajo a tiempo.

Gracias a mi familia por sus esfuerzos y ánimos.

Y gracias a mis amigos y compañeros que saben distraerte cuando lo necesitas. Un agradecimiento sincero a Sergio, Claudia, Patricia, Irene y Alberto.

1 Introducción

En este primer capítulo se desciben los motivos que llevaron a plantear este trabajo, los diferentes objetivos que se quieren alcanzar y, finalmente, se detalla la estructura que sigue el documento, con el fin de adelantar los diferentes elementos que se tratan en la memoria.

1.1. Planteamiento del problema

En los últimos años se ha experimentado, un gran incremento del número de dispositivos que cuentan con tecnología inalámbrica, acompañado del aumento de la cantidad y calidad de los recursos requeridos por cada usuario. Estos cambios han suscitado gran interés en la comunidad investigadora, que está aunando fuerzas para mejorar las prestaciones ofrecidas por las tecnologías inalámbricas.

Entre las opciones que han aparecido con objeto de mejorar la conectividad y el rendimiento, este trabajo se centra en el uso de Wireless Mesh Network (WMN), redes malladas inalámbricas, como medio de acceso a la red, y las técnicas de Network Coding (NC).

El uso de las WMN está totalmente extendido. Su crecimiento se debe principalmente a la flexibilidad que ofrecen. Sin embargo, el uso de un medio de transmisión inalámbrico se traduce en la aparición de numerosos inconvenientes. Estos van desde la seguridad de la

transmisión, al ser posible capturar el tráfico con mayor facilidad que en redes cableadas, hasta una gran pérdida de rendimiento. Este último factor es debido, en gran parte, a la existencia de interferencias.

Las técnicas de Network Coding, surgidas en la última década, pretenden paliar estos problemas. Se aprovechan de las características del medio inalámbrico para mejorar tanto el rendimiento como la seguridad de las comunicaciones. Sin embargo, hasta ahora no existe un claro análisis de las posibilidades de utilizar estas técnicas bajo escenarios de carácter aleatorio.

1.2. Objetivos

Son muchos los estudios que existen hasta el momento sobre el uso de técnicas de NC, pero siempre centrándose en las mejoras que ofrecen y, en la mayoría de los casos, sobre topologías canónicas sencillas. El principal objetivo de este trabajo es estudiar la aplicabilidad de NC sobre redes malladas inalámbricas.

Se pretende conocer la probabilidad que existe de utilizar NC sobre escenarios aleatorios. Para ello, se desarrolla un algoritmo que, dado un escenario, sea capaz de analizar, si es apropiado utilizar las técnicas de NC, por lo que primero se deberá estudiar bajo qué circunstancias es interesante, en términos de rendimiento, utilizar NC. En ningún momento se entra en el funcionamiento del protocolo subyacente aunque, por motivos obvios, es importante conocer el funcionamiento básico. Por otro lado, se estudiará la mejora del rendimiento que ofrece NC bajo escenarios desplegados aleatoriamente.

1.3. Estructura de la memoria

A continuación se explica la estructura del documento, revisando brevemente el contenido de cada uno de los capítulos.

- En el Capítulo 2 se presentan los conceptos previos necesarios para desarrollar el trabajo. Se definen aspectos como red mallada inlámbrica, escenarios sobre los que se trabaja; o sobre teoría de grafos, necesaria para desarrollar los algoritmos. También se realiza un acercamiento a los diferentes trabajos existentes centrados en el estudio de las técnicas de Network Coding, destacando la importancia de cada uno de ellos y la carencia que se pretende suplir con este trabajo.
- El Capítulo 3 supone la parte central del trabajo. En primer lugar se definen las condiciones necesarias para poder aplicar las técnicas de network coding sobre un

escenario y, posteriormente, se presentan dos algoritmos que permiten evaluar si un escenario cumple dichas condiciones. Cada uno de los algoritmos presenta una ventaja, pero ambos son necesarios.

- Con intención de analizar el rendimiento de Network Coding sobre topologías aleatorias se describe en el Capítulo 4 el simulador de redes utilizado, ns-3, y algunos desarrollos realizados sobre el mismo para poder llevar a cabo los análisis necesarios.

- En el Capítulo 5 se recogen dos tipos de resultados. Por un lado, se evalúa la probabilidad de poder utilizar Network Coding sobre despliegues aleatorios de red, aunque ni siquiera suponga rentable en términos de rendimiento. Por otro lado, los escenarios en los que se estima una mejora del rendimiento con estas técnicas se analizan con el simulador evaluando la mejora obtenida.

- Por último, en el Capítulo 6 se detallan las conclusiones que se han alcanzado tras el análisis de los resultados obtenidos previamente, tanto en la evaluación de la aplicabilidad de las técnicas de Network Coding como en la mejora en el rendimiento que ofrecen. Además, se presentan una serie de líneas futuras de investigación que han quedado abiertas tras realizar este trabajo.

2

Estado del Arte

Este capítulo recoge una introducción a cada uno de los elementos en los que se sustenta el trabajo. De esta forma el lector puede adquirir unos conocimientos básicos que le permitan comprender en mayor medida los diferentes conceptos tratados a lo largo de las siguientes páginas.

2.1. Redes Malladas Inalámbricas

Las redes malladas inalámbricas (Wireless Mesh Network, WMN) son una alternativa tecnológica para lo que se denomina *última milla* en el acceso a Internet, es decir, redes que actuan de puente entre el usuario y una puerta de salida a Internet (*gateway*). En los sistemas tradicionales la comunicación es siempre directa, desde los diferentes nodos a un único *gateway* y, por tanto, es necesario desplegar varios *gateways* para cubrir un gran área de cobertura. Sin embargo, las WMN permiten la conectividad directa entre los dispositivos hasta alcanzar el nodo de salida, como se muestra en la Figura 2.1, donde el nodo 15 podría actuar como *gateway*. Esto permite que cuando se quiera añadir un nodo a la red, este forma un papel activo en la misma, sin necesidad de llevar a cabo un rediseño. Sus ventajas son una gran escalabilidad y fiabilidad, debido a la existencia, en la mayoría de los casos, de múltiples alternativas de comunicación entre un nodo cualquiera y el *gateway*.

Sin embargo, su principal desventaja radica en la limitada capacidad que pueden ofrecer. A medida que el número de nodos aumenta, también lo hace el número de saltos

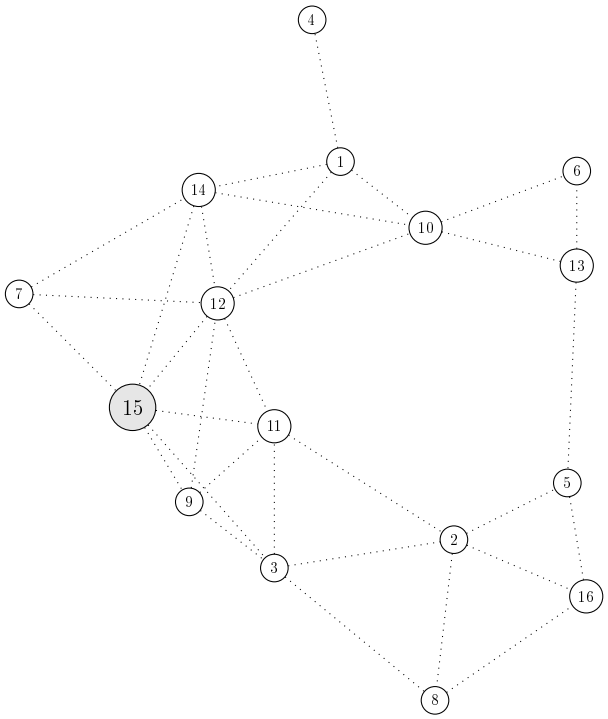


Figura 2.1: Ejemplo de red mallada inalámbrica

que hay que atravesar para llegar a una puerta de acceso a Internet, así como las posibles interferencias. Esto reduce notablemente la capacidad que puede ofrecer la red. Según algunos estudios [1], por cada nodo añadido, el rendimiento disminuye en $O(1/N)$. Otros trabajos [2] señalan el problema que supone el uso del protocolo TCP sobre este tipo de redes ya que fue diseñado para redes cableadas, donde la probabilidad de error es despreciable y la congestión es la principal causa de la pérdida de paquetes. Varias modificaciones de TCP han tratado de minimizar la pérdida de rendimiento que se produce sobre redes inalámbricas, destacando entre ellas las técnicas de Network Coding.

A la hora de trabajar con este tipo de redes se las suele representar como un grafo, en el que los vértices simbolizan los nodos inalámbricos y las aristas representan la conectividad directa entre dos nodos. Cada nodo tiene un área de cobertura, y para que exista una arista entre dos nodos, ambos deben de estar dentro del área de cobertura del otro.

2.2. Network Coding

Habitualmente, la información que transcurre a través de las redes de comunicación atraviesa los diferentes dispositivos intermedios sin ser tratada, únicamente retransmitida.

Cada uno de los flujos[1] de información 'viaja' por la red de manera independiente. Pero gracias a los avances tecnológicos se ha abierto la posibilidad de que los nodos por los que transcurren los flujos puedan procesar los paquetes recibidos, permitiendo gestionar los recursos de forma inteligente.

Se observó que era posible realizar operaciones sobre flujos independientes de datos para procesarlos conjuntamente y transmitirlos posteriormente. En concreto, se ha analizado la posibilidad de combinar en un nodo la información correspondiente a flujos independientes para que, posteriormente, sean los nodos destinos los que se encarguen de 'separar', decodificar, los datos que van realmente dirigidos a ellos, lo que podría resultar beneficioso para el rendimiento de las redes.

Network Coding es una técnica propuesta por Ahlswede et al. en [3]. Desde entonces, la comunidad investigadora ha realizado tremendos esfuerzos para conseguir aplicar estas técnicas sobre diferentes tipos de redes. Cabe destacar la gran labor investigadora que se está llevando a cabo en el caso de las redes inalámbricas, en concreto, sobre redes malladas. La idea principal de las técnicas de NC es combinar los paquetes que 'viajan' por la red de forman inteligente, consiguiendo mejorar el rendimiento ofrecido. A pesar de que pueden existir otro tipo de clasificaciones, podemos diferenciar entre dos métodos: aquellos que combinan paquetes pertenecientes a distintos flujos (*inter-flow*), frente aquellos que combinan paquetes del mismo flujo (*intra-flow*). En este trabajo se trabajará con técnicas *inter-flow* y, en este capítulo, se explicarán los principios básicos, algunas aplicaciones y ejemplos de esta técnica.

El esquema básico es el que se presenta a continuación.

- Uno de los nodos intermedios de la red, denominado *Coding Node*, se encarga de combinar, mediante codificación, paquetes procedentes de flujos independientes. Cada uno de los que formen el paquete codificado se denomina paquete nativo. Posteriormente, el nuevo paquete se transmite, utilizando la modalidad de *pseudo-broadcast*. Esta técnica consiste en dirigir el paquete a un único destino aunque lo puede recibir cualquiera dentro de un área de cobertura, aprovechando la naturaleza broadcast intrínseca al medio radio.
- Como, por norma general, los paquetes no llegarán al *Coding Node* de manera simultánea se introduce un temporizador con un valor *Coding Time* (C_T) que mantiene los paquetes en el *Coding Node* a la espera de otros posibles paquetes nativos.
- Otro parámetro a tener en cuenta es el *Buffer Size* (B_S), que determina cuantos paquetes se van a mantener a la espera simultáneamente en el *Coding Node*.
- En caso de sobrepasar el C_T o el B_S el primer paquete de ambos buffers se transmite según el esquema tradicional.

[1] Un flujo se refiere al intercambio de paquetes, comunicación, que existe entre una fuente y un destino

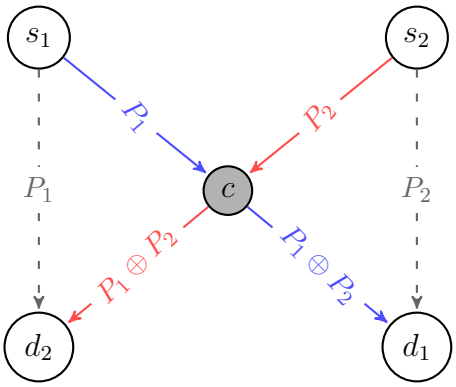


Figura 2.2: Escenario en X

- El último parámetro relevante es el *Overhearing Buffer*, que afecta a los nodos decodificadores que se mantienen a la 'escucha', almacenando posibles paquetes nativos que no vayan dirigidos a ellos y que podrían ser necesarios para la decodificación de paquetes deseados posteriormente.

En este trabajo la técnica de codificación utilizada es sencilla, consistente en realizar una operación XOR entre todos los paquetes nativos que conforman el paquete codificado. Para que el nodo destino pueda recuperar la información debe realizar, con el paquete codificado, las mismas operaciones XOR, utilizando los paquetes nativos que lo componen, excepto aquel que se pretende recuperar. Por tanto, el nodo destino tiene que ser capaz de almacenar en el *Overhearing Buffer* los paquetes nativos necesarios o, si no, se producirá lo que se conoce como un *fallo de codificación*.

Para ilustrar las técnicas de NC, la Figura 2.2 muestra una de las topologías canónicas utilizadas para mostrar sus beneficios. La red está compuesta por dos flujos, $s_1 \rightarrow d_1$ y $s_2 \rightarrow d_2$, que se intersecan en un nodo c que actúa de *Coding Node*.

Bajo el mecanismo tradicional si el nodo fuente, s_1, quiere enviar un paquete al nodo d_1, tiene que transmitirlo en primer lugar al nodo c para que éste lo retransmita al nodo d_1. De igual modo, si el nodo s_2 quiere enviar un paquete al nodo d_2, repetiría el mismo proceso. En total se realizan 4 transmisiones. Aplicando NC el nodo c almacenaría el primer paquete durante un periodo C_T, en caso de recibir un paquete del otro flujo aplicaría una operación XOR a ambos paquetes y retransmitiría, en modo *pseudo-broadcast*, un único paquete codificado ($P_c = P_1 \oplus P_2$) que recibirían d_1 y d_2. Estos dos, gracias a las características del medio inalámbrico, habrían recibido los paquetes P_2 y P_1, respectivamente y podrían recuperar los paquetes originales mediante las propiedades de la operación XOR: $P_1 = P_c \oplus P_2 = (P_1 \oplus P_2) \oplus P_2$ ó $P_2 = P_c \oplus P_1 = (P_1 \oplus P_2) \oplus P_1$.

Así, una comunicación que normalmente requiere de cuatro transmisiones, se puede completar con tan solo tres, haciendo uso de Network Coding. Se ha mejorado la eficiencia energética (menos retransmisiones) y se han reducido los retardos, los recursos ocupados y

las interferencias.

Volviendo a los estudios que existen sobre estas técnicas, cabe destacar la arquitectura propuesta por Katti et al. COPE [4]. Fue probablemente el primer trabajo en poner en práctica los conceptos teóricos recogidos en [[3],[5], [6], [7]]. En resumen, se simplifica todo el proceso de codificación y decodificación, empleando operaciones XOR a nivel de bit, tal y como describe en los párrafos anteriores.

Otro trabajo interesante es el llevado a cabo por Hunderboll et al. [8]. Ellos presentaron Coding Applied To Wireless On Mobile Ad-hoc Networks (CATWOMAN), técnica que fue concebida para operar sobre la solución Better Approach To Mobile Ad-hoc Networking (BATMAN) [9], integrando un completa solución de NC con ayuda de su popular protocolo de encaminamiento, usando los mensajes de gestión y control para identificar oportunidades de codificación y estimando la capacidad de 'escuchar' la información de los nodos vecinos. Sin embargo, los autores no consideran el uso de topologías complejas, evaluando el rendimiento de las soluciones sobre escenarios simples, en los cuales encontrar el *Coding Node* es inmediato.

En el Grupo de Ingeniería Telemática de la Universidad de Cantabria se han realizado varios estudios [[10], [11]] sobre el rendimiento de las técnicas de NC sobre topologías canónicas, como la que aparece en la Figura 2.2. Estos trabajos se centran en el impacto de los parámetros operacionales (*Coding Time*, *Overhearing Buffer*, *Coding Buffer*). Aunque los resultados que se obtienen presentan una mejora relevante sobre canales ideales, es decir, cuando todos los paquetes llegan a sus destinos, en el momento en que los canales inalámbricos tienen una cierta probabilidad de error, la mejora ofrecida por NC no es la que se podría esperar, reflejando una gran pérdida de rendimiento frente a la que se observa al emplear las técnicas tradicionales TCP.

Uno de las características más importantes de las redes WMN es su estructura aleatoria, ya que la posición y movilidad de los nodos pueden ser impredecibles. En tales casos, identificar un *Coding Node* y, en concreto, una solución óptima para llevar a cabo la codificación resulta complejo. Por eso surgen estudios que intentan proporcionar una serie de mecanismos para identificar al apropiado *Coding Node* a lo largo del escenario. Por ejemplo, Le et al. [12] introducen Distributed Coding-Aware Routing (DCAR), un protocolo de encaminamiento bajo demanda que consigue descubrir posibles *Coding nodes*; también presentan Coding-aware Routing Metric (CRM), una métrica que permite comparar las configuraciones que proporcionan mayores beneficios. Posteriormente, Gou et al. [13] cuestionan si los requerimientos propuestos son suficientes cuando existen varios nodos de intersección entre flujos, y proponen una nueva métrica Free-ride Oriented Routing Metric (FORM), capaz de encontrar oportunidades de codificación, sin importar el número de flujos y nodos de intersección. Ambos artículos son de gran importancia en el trabajo que se está exponiendo, puesto que se parte de las condiciones de codificación que presentan.

2.3. Teoría de Grafos

En esta sección se recogen los principales conceptos sobre teoría de grafos utilizados durante el trabajo. La necesidad de comprender la teoría de grafos es obvia. Para poder desarrollar el algoritmo con los objetivos marcados en la Sección 1.2 se trabaja la red como un grafo dirigido donde los nodos que componen la red son los vértices del grafo y las aristas conectan aquellos nodos con conectividad física entre ellos.

2.3.1. Conceptos básicos

Primero se define el concepto de grafo dirigido

DEFINICIÓN **2.1.** *Un grafo es un par ordenado* $G = (V, E)$*, donde*

- $V \neq \emptyset$*, un conjunto no vacio de objetos llamados vértices o nodos*
- $E \subset \{(a, b) \in VxV : a \neq b\}$*, conjunto de pares ordenados de elementos de* V *denominados aristas o arcos, donde por definición una arista va del primer elemento (a) al segundo (b) dentro del par.*

Para poder representar un grafo se trabaja con la matriz de adyacencia.

DEFINICIÓN **2.2.** *Matriz de adyacencia es una matriz de dimensión* $N \times N$ *y cada elemento de la matriz* $a_{ij} \in 0, 1$

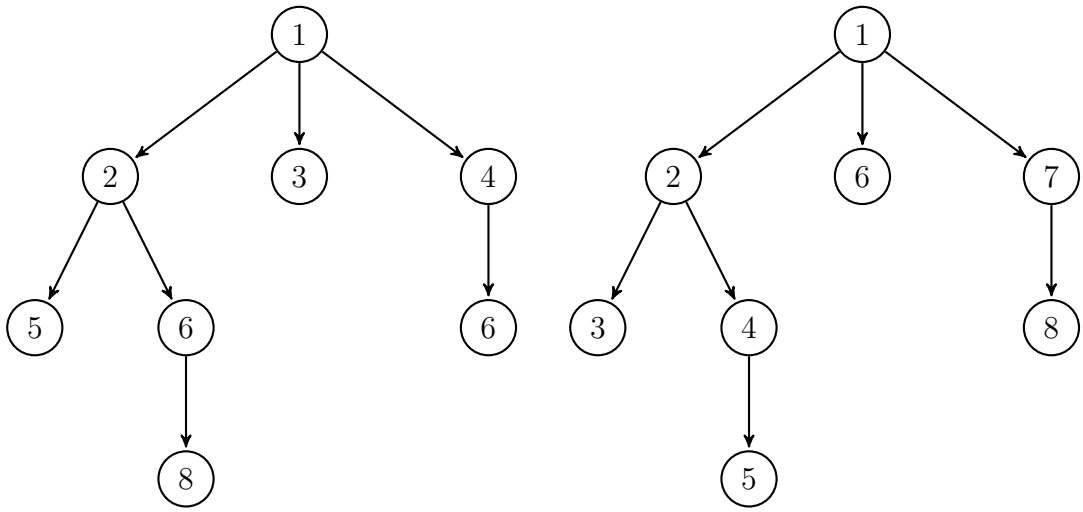
Esta matriz representa a un grafo, donde N *es el número de vértices y si un elemento* $a_{ij} = 1$ *el vértice* i *es adyacente al vértice* j *en esa dirección.*

La matriz de adyacencia es una representación que resulta útil para los algoritmos que requieren conocer la existencia de aristas entre dos vértices. Además, se puede generalizar la matriz de forma que se puede asignar un peso p diferente a cada arista, entre los vértices i, j, estableciendo el elemento $a_{i,j} = p$

2.3.2. Algoritmos de exploración de grafos

La solución propuesta hace uso de algunos algoritmos básicos, necesarios para varias operaciones de exploración. A continuación se citan los más conocidos.

- **Breadth-First Search (BFS):** es un conocido algoritmo de exploración de grafos, que comienza en un nodo raíz para después recorrer uno por uno todos los vértices



(a) BFS (b) DFS

Figura 2.3: Diferentes algoritmo de exploración de grafos

adyacentes. Se define como un algoritmo de búsqueda en 'anchura' puesto que primero explora todos los vértices de un mismo nivel antes de pasar al siguiente. En la Figura 2.3a se observa un ejemplo de exploración de un grafo sencillo. Su eficiencia espacial usando matrices de adyacencia es $O(|V|)$, mientras que la temporal es $O(|V| + |E|)$.

- **Depth-Firts Search (DFS):** otro conocido algoritmo de exploración que, en este caso, se explora el grafo en ' 'profundidad'. Comienza en un vértice raiz y va recorriendo el grafo por los vértices adyacentes de manera recursiva hasta llegar a un vértice *hoja*, es decir que no tiene más nodos adyacentes. Se puede ver en la Figura 2.3b un ejemplo ilustrativo. La complejidad espacial es $O(|V|)$, y la temporal $O(|E|)$ (salvo casos concretos).

2.3.3. Algoritmos para hallar conectividad en el grafo

Definición **2.3.** *Un grafo $G = (V, E)$ se denomina grafo conexo, si para cada par de vértices en el grafo existe al menos una sucesión de vértices adyacentes entre ellos.*

Es importante conocer si un grafo dado es conexo, puesto que parte de los algoritmos utilizados requieren que así sea. Visto desde el punto de vista de las redes de comunicación, que un grafo sea conexo significa que cualquier par de nodos de la red son capaces de establecer una comunicación, entre ellos; cuando un conjunto de nodos no puede comunicarse con otro se tendrían en realidad dos redes independientes.

Por tanto, es necesario contar con un algoritmo que permita decidir si un grafo es

conexo o no. Este algoritmo utiliza una búsqueda DFS y la matriz de adyacencia. Su funcionamiento básico es el que se presenta a continuación.

- Se comienza por un nodo raíz r y un vector con el número de vértices del grafo incializado a 0
- Se explora el árbol con el algoritmo DFS; cada vez que se recorre un vértice se establece, en la posición del vector correspondiente a ese vértice, un 1.
- Una vez recorrido el árbol, si el vector no contiene ningún 0, entonces el grafo es conexo. En caso contrario el grafo es no conexo.

La complejidad de este algoritmo es la misma que la del algoritmo de exploración DFS.

2.3.4. Algoritmos de búsqueda de la ruta más corta

Los algoritmos que permiten encontrar las rutas más cortas son la base de la solución propuesta. Asignando un peso a cada arista del grafo, el problema consiste en encontrar el conjunto de aristas entre dos vértices cualesquiera del grafo con el menor peso global, entendiendo peso global como la suma de pesos de cada una de las aristas que conforman el camino. Si este peso se establece a 1 para todas las aristas, el peso global o longitud de un camino, será igual al número de saltos, o aristas, de dicho camino.

- **Dijkstra:** es uno de los algoritmos más conocidos de la teoría de grafos, propuesto por Edger Dijkstra en 1959 [14]. Dado un grafo conexo con pesos en la aristas no negativos, encuentra el camino de peso mínimo entre un vértice dado y el resto de vértices del grafo. La salida del algoritmo es un árbol generador de peso mínimo. La complejidad del algoritmo es $O(|V|^2)$, se puede mejorar la complejidad del algoritmo utilizando colas de prioridad a $O((|E|+|V|)*log(|V|))$. El funcionamiento básico es:

  1. Inicializar un vector de distancias, a un valor 'infinito', excepto la del vértice s que se establece a 0.
  2. Recorrer todos los vértices adyacentes al vértice a, siendo $a = s$ en la primera iteración
  3. Si el peso P desde s hasta v_i es menor que el almacenado en el vector de distancias, entonces se sustituye por éste en el vector.
  4. Se marca el vértice a y se establece a con el vértice con menor distancia entre los no marcados.

5. Se repite desde el paso 3, mientras existan vértices no marcados.

- **K-shortest Path:** es otro algoritmo ampliamente usado, que permite encontrar K caminos de menor peso sin ciclos entre dos vértices dados de un grafo. El algoritmo con menor complejidad y el utilizado en este trabajo es el presentado por Jin Y. Yen en 1971 [15]. El funcionamiento a grandes rasgos es el siguiente:

  1. Se encuentra el camino de menor coste, mediante el algoritmo de Dijsktra, entre el vértice origen s y destino d.
  2. Se Eliminan las aristas del grafo según un criterio establecido para calcular rutas alternativas entre s y d, evitando crear ciclos. Se Almacenan las rutas obtenidas temporalmente
  3. Se Repite el paso 2 un número de veces equivalente al número de saltos del último camino más corto.
  4. Se selecciona el camino con menor peso entre los almacenados temporalmente.
  5. Se repite $K-1$ veces desde el paso 2.

  La complejidad de este algoritmo es dependiente del algoritmo usado para el cálculo del camino de peso mínimo; en este caso, como ya se ha indicado, se utilizará Dijkstra. El algoritmo llama al algoritmo de Dijsktra $K*l$ veces, siendo l el número de vértices de lo que se denomina durante el algoritmo *spur paths*, rutas entre diferentes vértices y el vértice destino, y su valor esperado es $O(log|V|)$ y en el peor caso $|V|$. La complejidad en el peor caso es $O(K*|V|^2*(|V|^2+|E|)*log(|V|))$, siendo K el número de caminos mínimos que se pretenden hallar.

3

Algoritmo implementado

En este capítulo se detallará el algoritmo propuesto para cumplir los objetivos planteados previamente. Primero se explicarán varios conceptos con los que se trabaja. También se presentarán otras funciones que son necesarias para poder obtener los resultados analizados en el Capítulo 5.

3.1. Condiciones de codificación

Como ya se comentó en la Sección 2.2 Le et al. exponen en [12] las condiciones que debe cumplir un nodo de la red para poder actuar como *Coding Node*. El algoritmo desarrollado parte de estas condiciones, y devuelve la lista de posibles *Coding Nodes* (si existe alguno) sobre cualquier red inalámbrica.

Antes de presentar las condiciones de codificación se deben formalizar algunos conceptos:

- Se denomina a a un nodo cualquiera de la red.
- Se define $\mathcal{N}(a)$ como el conjunto de nodos adyacentes al nodo a.
- Si F es un flujo [1], entonces $a \in F$ denota que el nodo a pertenece al conjunto F.

[1] Se utiliza flujo o camino indistintamente

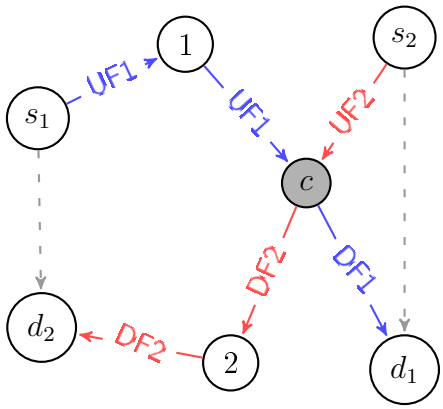


Figura 3.1: Ejemplo de esquema de codificación

- Se define $\mathcal{U}(a, F)$ al conjunto de nodos *upstream*, formado por aquellos entre el origen del flujo F y a.
- Se define $\mathcal{D}(a, F)$ al conjunto de nodos *downstream*, formado por aquellos entre a y el destino del flujo F.
- Se define el nodo de articulación c si dado dos flujos F_1 y F_2, c pertenece a ambos. Estos son los potenciales *Coding Nodes*.

Para entender bien los conceptos detallados se muestra un ejemplo en la Figura 3.1, donde se tienen dos flujos de comunicación $F_1 = s_1 \to d_1$ y $F_2 = s_2 \to d_2$. En este ejemplo se ve que $\mathcal{U}(c, F_1) = \{s_1, 1\}, \mathcal{D}(c, F_1) = \{d_1\}, \mathcal{U}(c, F_2) = \{s_2\}$ y $\mathcal{D}(c, F_1) = \{2, d_2\}$. Además, el nodo c es un nodo de articulación y, por tanto, es un potencial *Coding Node*. El nodo c puede actuar de *Coding Node* si y sólo si cumple las condiciones de codificación.

DEFINICIÓN **3.1.** *Las condiciones de codificación para dos flujos, F_1 y F_2, que intersecan en el nodo c son:*

1. *Existe un nodo $u_1 \in \mathcal{D}(c, F_1)$ tal que $u_1 \in \mathcal{N}(v_2)$, $v_2 \in \mathcal{U}(c, F_2)$, o $u_1 \in \mathcal{U}(c, F_2)$.*
2. *Existe un nodo $u_2 \in \mathcal{D}(c, F_2)$ tal que $u_2 \in \mathcal{N}(v_1)$, $v_1 \in \mathcal{U}(c, F_1)$, o $u_2 \in \mathcal{U}(c, F_1)$.*

TEOREMA **3.1.** *Si se asume condiciones ideales en el medio inalámbrico de comunicación, es decir, que todos los paquetes enviados llegan correctamente a su destino, las condiciones descritas en 3.1 son necesarias y suficientes para que se produzcan una codificación y decodificación correcta.*

En otras palabras, estas condiciones aseguran que los nodos destino son capaces de 'escuchar' los paquetes nativos correspondientes al flujo contrario y poder así decodificar el paquete codificado que recibirán del nodo c. Los autores de [13] cuestionan que estas

condiciones sean válidas si existen varios nodos de articulación entre dos pares origen-destino, proponiendo unas condiciones generalizadas.

Para poder operar según el protocolo presentado en [[10],[11]] es necesario realizar ligeros cambios sobre las condiciones de la Definición 3.1. Se añade una restricción al nodo que se encarga de decodificar los paquetes codificados, que debe coincidir el nodo destino del flujo correspondiente. Además, únicamente se va a trabajar con dos flujos de datos y un único nodo codificador. Considerando estas limitaciones se pueden reescribir las condiciones descritas en la Definición 3.1 tal y como se ve a continuación:

DEFINICIÓN **3.2.** *Las condiciones simplificadas de codificación para dos flujos, F_1 y F_2, que intersecan en el nodo c son:*

1. *Existe un nodo $u_1 \in \mathcal{U}(c, F_1)$ tal que $u_1 \in \mathcal{N}(d_2)$ y d_2 es el nodo destino de F_2, o $u_1 = d_2$.*
2. *Existe un nodo $u_2 \in \mathcal{U}(c, F_1)$ tal que $u_2 \in \mathcal{N}(d_1)$ y d_2 es el nodo destino de F_1, o $u_2 = d_1$.*

TEOREMA **3.2.** *Si se asume condiciones ideales en el medio inalámbrico de comunicación, las condiciones anteriores son suficientes para que se produzca una codificación y decodificación correctas.*

TEOREMA **3.3.** *Si a las condiciones del Teorema 3.2 se les añade que únicamente los nodos destino pueden ser los decodificadores, estas condiciones pasan a ser suficientes y necesarias.*

DEMOSTRACIÓN **3.1.** *Para asegurar que los destinos de ambos flujos puedan llevar a cabo la decodificación deben ser capaces de escuchar los paquetes nativos que conforman el paquete codificado. Para ello cada nodo destino tiene que ser adyacente a algún nodo del conjunto* upstream *del flujo contrario, o pertenecer a dicho conjunto. Para que además sean condiciones suficientes se tiene que garantizar la condición ideal del canal inalámbrico, así siempre se podrá decodificar el paquete codificado, si no se producen errores.*

Se ha mencionado en varias ocasiones el hecho de tener canales ideales de transmisión. También se puede aplicar las condiciones de codificación sobre escenarios reales, donde existe cierta probabilidad de error entre los enlaces de comunicación, pero no sería posible asegurar que en todas las ocasiones los procesos de codificación y decodificación sean posibles. Se pueden producir, según las probabilidades de error del canal, ciertos *fallos de decodificación*. Por esto, se sugiere que, a la hora de seleccionar los nodos adyacentes a otro, se haga sobre aquellos que tengan una conexión con una baja probabilidad de error. Que aparezcan fallos de decodificación, debido a que no se han podido 'escuchar' los paquetes nativos, puede suponer una pérdida del rendimiento, empeorando los resultados que se obtienen usando el protocolo TCP tradicional.

Volviendo a la Figura 3.1, es claro que cumple las condiciones citadas. El nodo $d_2 \in N(s_1)$ y $s_1 \in \mathcal{U}(c, F_1)$ y, por otro lado, $d_1 \in N(s_2)$ y $s_2 \in \mathcal{U}(c, F_2)$. Luego se puede afirmar que el nodo c cumple las condiciones de codificación, y que se podrían aplicar las técnicas de NC; aunque se respeten estas condiciones, no asegur que se vaya a obtener un mayor rendimiento utilizando NC, por lo que es algo a evaluar posteriormente.

3.2. Primer Algoritmo

Una vez que se han establecido las condiciones necesarias (así como las restricciones impuestas por el protocolo NC disponible) que un escenario debe cumplir para poder utilizar las técnicas de NC, se presenta el Algoritmo 1, que permite establecer si un escenario es apropiado para aplicar técnicas de NC, y qué nodos son los más apropiados para ser *Coding Nodes*.

Para entender mejor el algoritmo cabe destacar que en primer lugar siempre se calcula las rutas obtenidas mediante el algoritmo de Dijkstra entre los dos pares de nodos fuente/destino. Estas rutas serían las utilizadas bajo TCP, y no tienen porque coincidir con las necesarias para utilizar NC, incluso siendo estas de mayor longitud en algunas ocasiones. La idea del algoritmo es calcular, por cada nodo de la red, las K-rutas más cortas desde ese nodo hacía los cuatro nodos (s_1,s_2,d_1,d_2) y comprobar si se cumplen las condiciones de codificación simplificadas. El número de rutas calculadas, K, es un parámetro del algoritmo.

El algoritmo recibe como entradas un grafo conexo $G = (V, E)$, que representa el escenario, donde V es el conjunto de nodos de la red y E es el conjunto de aristas/enlaces entre los nodos. Además, también necesita dos pares de nodos s_1/d_1,s_2/d_2 que establecen los correspondientes flujos, F_1 y F_2. Por último, se utiliza un parámetro, Δ, que determina el máximo número de saltos adicionales que un camino podría necesitar para encontrar un *Coding Node*, frente al flujo con mayor número de saltos en TCP. El algoritmo devuelve la lista de potenciales *Coding Nodes* que satisfacen las condiciones de la Definición 3.2. Para trabajar posteriormente con los escenarios en el simulador también se almacenan las rutas que se obtienen para utilizar NC sobre el escenario, aunque esto no se refleje en el algoritmo.

La complejidad del algoritmo estudiado es, claramente, dependiente de los algoritmos que se empleen para el calculo de caminos mínimos. En este caso, se parte de las complejidades expuestas en el Capítulo 2.3.2 para el algoritmo de Dijkstra ($O((|E|+|V|)*log(|V|))$) y el presentado por Jin Y. Yen ($O(K*|V|*(|V|+|E|)*log(|V|))$). El algoritmo calcula, en el peor de los casos, para cada nodo K caminos mediante el algoritmo presentado por Yen. Por tanto, la complejidad en el caso peor es $O(K*|V|^2*(|V|+|E|)*log(|V|))$.

El principal problema del algoritmo es que no asegura encontrar todas las posibles

Data: $G(V,E), s_1, s_2, d_1, d_2, \Delta$
Result: Lista de posibles nodos codificadores
Calcular F_1 and F_2 with Dijkstra;
for *each node* i *in* $G \neq s_1$, s_2, d_1 *and* d_2 **do**
  Calcular $\mathcal{U}_K(i,F_1)$, $\mathcal{D}_K(i,F_1)$, $\mathcal{U}_K(i,F_2)$, $\mathcal{D}_K(i,F_2)$ mediante el Yen's Algorithm;
  for $j = 1 : K$ **do**
    if $(Length(\mathcal{U}(i,F_1)+\mathcal{D}(i,F_1)) \leq Length(F_1)+\Delta)$ *and* $(Length(\mathcal{U}(i,F_2)+\mathcal{D}(i,F_2)) \leq Length(F_2)+\Delta)$ **then**
      $LU1 = [\,]$;
      for *each node,* n, *in* $\mathcal{U}_j(i,F_1)$ **do**
        $LU1 = [LU1\ n\ N(n)]$;
      end
      $LU2 = [\,]$;
      for *each node,* n, *in* $\mathcal{U}_j(i,F_2)$ **do**
        $LU2 = [LU2\ n\ N(n)]$;
      end
      if $(d_1 \in LU2)$ *and* $(d_2 \in LU1)$ **then**
        $CodingNodes = [CodingNodes\ i]$;
      end
    end
  end
end

Algoritmo 1: Pseudo-código para comprobar si existen potenciales *Coding Nodes* en una topología multi-salto

oportunidades, siendo dependiente del parámetro K. Existen varias rutas entre el nodo bajo análisis y los distintos nodos origen y destino, por lo que es posible que con alguna ruta no se cumplan las condiciones de codificación, pero con otra combinación de rutas sí. Para ejemplificar este problema se presenta la Figura 3.2. Si el algoritmo de encaminamiento escoge las cuatro rutas con línea continua el algoritmo devolverá que no se pueden aplicar las técnicas de NC, puesto que el nodo d_2 no es vecino de ningún nodo del camino upstream (s_1 y 1), a pesar de que d_1 sí es vecino del nodo s_2. Sin embargo, si se escoge como rutas alternativas las representadas con flechas discontinuas, el nodo d_2 pertenece al conjunto de nodos adyacentes de 3 y se cumplen, por tanto, las condiciones de codificación.

A pesar de que el algoritmo presenta una clara dependencia con el parámetro K, tal y como se verá en el Capítulo 5, los resultados son casi idénticos para cualquier valor de K en los escenarios tratados. Podría suponer un problema sobre redes en las que el número de rutas alternativas con el mismo número de saltos, por ejemplo para densidades altas de nodos.

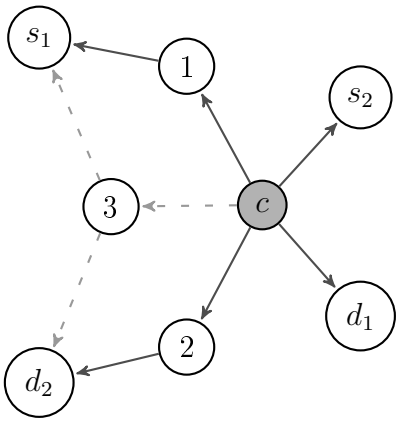


Figura 3.2: Ejemplo de esquema de codificación. Problemática del primer algoritmo

3.3. Algoritmo exacto

Se verá más adelante que, las técnicas de Network Coding pierden interés cuando el número de saltos en alguno de los flujos es mayor que el que sería necesario utilizando TCP. Por ello, se presenta el Algoritmo 2, que soluciona la dependencia con el parámetro K, pero limita el parámetro $\Delta = 0$.

Se hace un primer descarte de aquellos nodos que no alcanzan a los diferentes nodos origen y destino en el mismo número de saltos que las rutas originales y, para comprobar las condiciones de codificación, se basa en el número de saltos mínimo y no en las rutas calculadas. Es decir, para comprobar si el nodo d_1 pertenece al conjunto $\mathcal{U}(c, F_2)$ se calcula la longitud del camino $s_2 \to c$, y se compara con la longitud del camino $s_2 \to d_1 + d_1 \to c$, obtenidas ambos por el algoritmo de Dijkstra. En caso de ser iguales, existe una ruta $s_2 \to d_2 \to c$ de igual longitud que $s_2 \to c$. Se continua el proceso con todos los vecinos de d_1 y con el otro nodo destino. Si se cumple para algún nodo entre el conjunto formado por $d_1 \cup \mathcal{N}(d_1)$ y para otro, del conjunto formado por $d_2 \cup \mathcal{N}(d_2)$, ese nodo c puede actuar como un *Coding Node*.

La principal ventaja de este algoritmo es que, como se ha comentado, su resultado no depende de ningún parámetro. Es decir, gracias a este algoritmo se puede comprobar cuántos escenarios de carácter aleatorio podrían utilizar técnicas de NC. Además, al añadir la limitación en el número de saltos entre las rutas resultantes frente a las originales es bastante razonable pensar que se mejorará el rendimiento de la red. Sin embargo, con este algoritmo no se obtienen los caminos que se deben configurar para utilizar las técnicas correctamente. Habría que incorporar la funcionalidad necesaria para encontrar las rutas que permiten utilizar NC, basándose, por ejemplo, en un algoritmo para encontrar las K-rutas más cortas y ver si entre ellas están las buscadas y, si no, aumentar el valor de K.

La complejidad de este segundo algoritmo viene marcada por la búsqueda de la ruta

```
Data: G, s1, s2, d1, d2
Result: List of possible coding nodes
Calculate F1 and F2 with Dijkstra ;
for each node i in G ≠ s1, s2, d1, d2 do
    Calculate U(i,F1), D(i,F1), U(i,F2), D(i,F2) with Dijkstra Algorithm ;
    if (Length(F1) = Length(U(i,F1)) + Length(D(i,F1))) and (Length(F2) =
    Length(U(c,F2) + D(c,F2))) then
        LN1 = [N(d1) d1] ;
        LN2 = [N(d1) d2] ;
        for each node j in LN1 do
            Calculate path j → s2 ;
            Calculate path j → i ;
            if (Length(j → s2) + Length(j → i) == Length(U(i,F2))) then
                for each node m in LN2 do
                    Calculate path m → s1 ;
                    Calculate path m → i ;
                    if (Length(j → s1) + Length(j → i) == Length(U(i,F1)))
                    then
                        CodifingNodes = [CodifingNodes i] ;
                    end
                end
            end
        end
    end
end
```

Algoritmo 2: Pseudo-código para comprobar si es posible utilizar Network Coding

mediante el algoritmo de Dijkstra en las diferentes etapas. Para cada vértice se aplica cuatro veces Dijsktra; en caso de cumplir las condiciones establecidas se vuelve a aplicar Dijsktra dos veces y, por último, en caso de que se vuelvan a cumplir ciertas condiciones establecidas se aplicaría nuevamente en dos ocasiones Dijsktra. En el caso peor esas condiciones se cumplirán siempre, y la complejidad sería $O(|V|*(D+(|V|*(D+|V|*D))))$ $O(|V|^3*D^2)$, siendo D la complejidad de Dijsktra $O((|E|+|V|)*log(|V|))$

4

Network Simulator

Con objeto de comparar el rendimiento de los escenarios bajo el esquema tradicional de TCP frente a NC se exportan los escenarios generados a un simulador de redes. Este capítulo introduce dicho simulador, ns-3. Se describe sus principales características y algunos pasos previos a la campaña de simulación que se presentará con el capítulo siguiente.

4.1. Características

El simulador se comienza a desarrollar bajo el nombre de *REAL*, por Srinivasan Keshav en el año 1989. Gracias a la implementación llevada a cabo por Lawrance Berkely National Laboratory (LBNL) entre los años 1995-97, apareció la primera versión conocida como ns-1. En ella el núcleo estaba escrito en C++, y se apoyaba en Tool Command Language (Tcl) para la descripción de los escenarios a simular. Durante el periodo de vida de esta primera versión se unieron contribuciones de Sun Microsystems y UC berkeley, entre otros. Entre los años 1996-97 surgió la segunda versión del proyecto, conocida como ns-2. Se mantuvo el núcleo en C++, pero se sustituyó el uso de Tcl por Object Tool Command Languaje (OTcl), una versión orientada a objetos de Tcl. Finalmente, en los años 2004-05 se comenzó a trabajar en la última versión conocida, ns-3, bajo el trabajo liderado por Tom Henderson de la Universidad de Washintong. Se abandonó por completo la compatibilidad hacia atrás con ns-2, debido a la alta sobrecarga de mantenimiento que podría acarrear. En junio de 2008 se lanzó la primera versión, ns-3.1 y se sigue en constante proceso de renovación, encontrándose actualmente por la versión ns-3.20, publicada en junio de 2014.

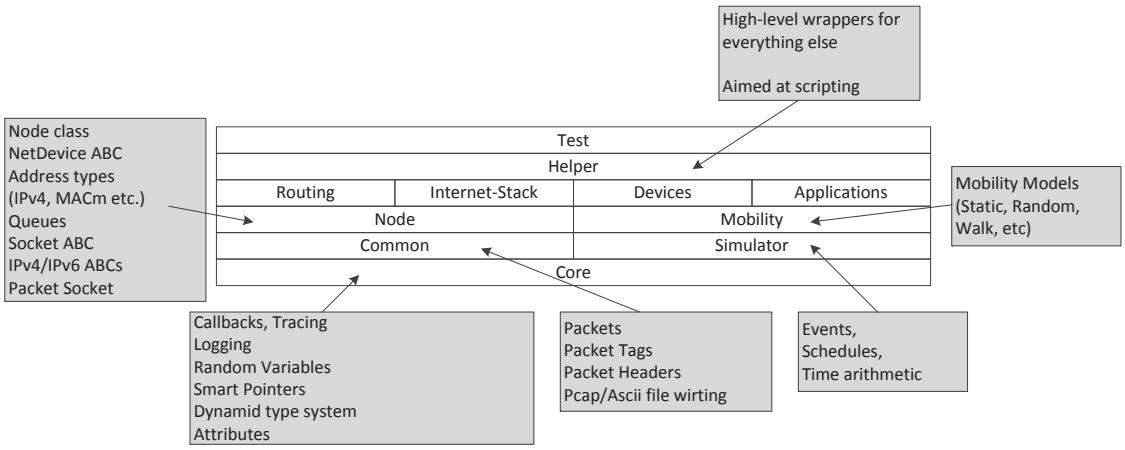


Figura 4.1: Organización software de ns-3

Aunque como se ha mencionado, ya existe la versión ns-3.20, en este proyecto se hace uso de la versión ns-3.13, que apareció en diciembre de 2011. El principal motivo es trabajar con los módulos desarrollados para esta versión por el Grupo de Ingeniería Telemática de la Universidad de Cantabria. Además, el simulador no cuenta con un módulo operativo para trabajar con *network coding* y es necesario utilizar uno desarrollado a tal efecto. Ese mismo módulo ha sido utilizado en trabajos previos como los citados en Capítulo 2 [[11], [10]].

Uno de los grandes atractivos de ns-3 es su carácter de software libre, lo que ha facilitado su uso masivo en diferentes sectores, como son la industria, entornos académicos o, incluso, gobiernos. Cabe destacar también su flexibilidad y su carácter modular que, en labores de investigación, brinda la posibilidad de incorporar nuevos componentes con funcionalidades que anteriormente no estaban implementadas. Aunque su manejo es tedioso y complejo, puesto que requiere un conocimiento profundo de su arquitectura interna, las opciones de simulación que brinda son muy amplias, permitiendo realizar estudios a gran escala, que hoy en día son absolutamente imprescindibles.

ns-3 es un simulador de redes de eventos discretos, en el que tanto el núcleo como los modelos se implementan en C++. De forma simplificada, y para no entrar en complicaciones que se alejan del objetivo de este trabajo, se cuenta con un programa C++ principal. Este define la topología de la simulación y enlaza con las diferentes librerías con las que cuenta ns-3, dando lugar al escenario que se desea simular. También se describen en el programa principal, las características básicas de la simulación, desde la cantidad de información transmitida al tiempo de la conexión. Además, ns-3 exporta la mayoría de sus APIs a Python, permitiendo crear escenarios de simulación también en este lenguaje.

El código fuente de ns-3 está mayormente organizado en el directorio *src* y se estructura como refleja la Figura 4.1. Generalmente las dependencias de los módulos se limitan a aquellos con los que están conectados directamente. Es importante, a la hora de hacer uso

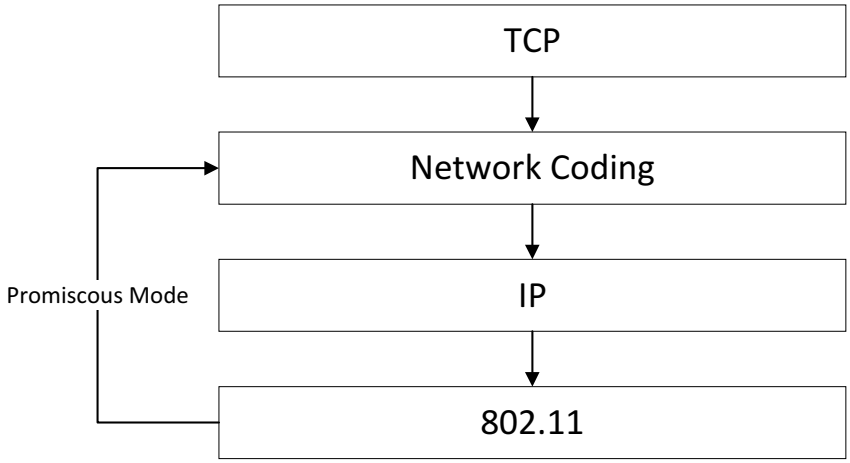


Figura 4.2: Modelo de Capas

de esta herramienta, tener claro la estructura de cada uno de los módulos, ya que facilita sobremanera su control.

4.2. Network Coding

El simulador ns-3 va evolucionando, pero a día de hoy no soporta el uso de *network coding*. Pero gracias a su versatilidad y carácter modular, en el Grupo de Ingeniería Telemática ha desarrollado un objeto que implementa las funcionalidades descritas en la Sección 2.2.

En concreto, se cuenta con un módulo que se sitúa entre las instancias de IP y TCP en el modelo de capas de Internet, actuando como una capa independiente intermedia. Hace uso del modo *promiscuo* del nivel de enlace 802.11 para 'escuchar' todos los paquetes que recibirá el nodo, a pesar de que no vayan dirigidos a él, y actuar en consecuencia. Es decir, si se trata de un nodo destino, almacenará los paquetes nativos en el *Overhearing Buffer* o, si se trata de un nodo codificador, llevará a cabo las operaciones correspondientes de almacenamiento, codificación o retransmisión.

De manera sencilla, cada vez que un nodo capta un paquete puede ser por: ir dirigido a él a nivel IP, siendo procesado por la función *ForwardingReception*, o no, supuesto en el que lo gestionará la función *ReceivePromiscous*. A continuación se presenta un esquema general.

ReceivePromiscous:

- Si el paquete no está codificado, y el nodo en cuestión es destino, se actualiza el *OverhearingBuffer*.

- Si el paquete está codificado:
  - Se comprueba la cabecera NC para ver si el nodo es receptor de ese paquete.
  - En caso afirmativo, se intenta decodificar el paquete y, si fuera posible, se pasa al nivel superior TCP.
  - En caso contrario se descarta.

ForwardingReception:

- Si el nodo en cuestión es un *CodingNode*, actualiza el *CodingBuffer* y busca una oportunidad de codificar flujos.
- En caso contrario se retransmite el paquete, en la manera tradicional al siguiente nodo de la ruta.

Según el diseño previo el nodo destino solo puede situarse a un salto del *Coding Node*. Una vez que el paquete es codificado en el *Coding Node*, este transmite el paquete en modo *psuedo-broadcast*, por lo que lo reciben todos los nodos adyacentes, aunque realmente solo va dirigido a uno de ellos, siendo el siguiente nodo de la ruta de uno de los dos flujos. El paquete codificado va dirigido, por defecto, al destino que corresponde al primer paquete nativo. El problema surge cuando dos flujos se bifurcan, es decir, existirán dos nodos que reciban el paquete codificado pero sólo uno lo retransmitirá al siguiente salto, mientras que el otro, lo descartará, a no ser que sea el nodo destino de uno de los flujos.

Esta limitación es probablemática y, por tanto, se ha manipulado el código del módulo de NC. Por no replantear todo el esquema creado se establece un parámetro de configuración a los nodos adyacentes al *Coding Node* con el identificador del nodo destino del flujo al que pertenecen. Además, se ha alterado la función *ReceivePromiscous* con el siguiente esquema:

- Si el paquete no está codificado, y el nodo es cuestión es destino, se actualiza el *OverhearingBuffer*
- Si el paquete está codificado:
  - Se comprueba la cabecera NC para ver si el nodo es receptor de ese paquete.
  - En caso afirmativo, se intenta decodificar el paquete y, si fuera posible, se pasa al nivel superior TCP.
  - En caso contrario, si el es vecino del *CodingNode* se comprobará si la dirección IP destino es la correspondiente a su flujo, en caso contrario, se retransmitirá al siguiente nodo de la ruta, modificando la cabecera IP, sustituyendo la dirección destino por la correspondiente al nodo destino del flujo en cuestión.

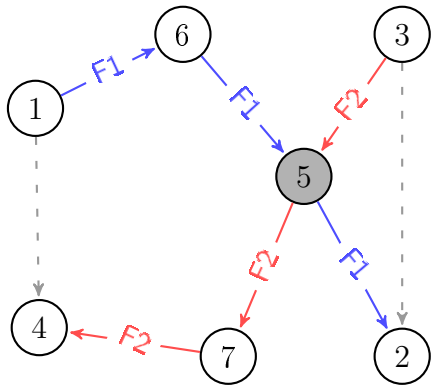


Figura 4.3: Resultado de la configuración de los nodos según los ficheros de configuración

De esta forma no se altera el esquema de NC y permite que es comparar el rendimiento que se obtiene según los dos esquemas NC y TCP.

4.3. Exportar Escenarios

Como se ha adelantado perviamente el código desarrollado debe generar una serie de ficheros para poder configurar los escenarios en el marco ns-3. En concreto se necesitan cuatro ficheros: configuración del escenario, configuración del canal, configuración de las rutas y configuración de los parámetros de simulación. A continuación se expondrán cada uno de ellos.

La posición de los nodos se define según aparece en la Tabla 4.1, incluye el número del nodo, su posición según las coordenadas x, y, z, si se trata de un nodo transmisor (TX), receptor (RX), si es un *CodingNode* (CR) y, por último, si es un nodo vecino del *CodingNode* (NCR). En caso de que sea un nodo transmisor o receptor se establece qué otro nodo es el otro extremo de la conexión correspondiente. El resultado de la configuración que se recoge en la Tabla 4.1 es la topología que se muestra en la Figura 4.3.

Tabla 4.1: Fichero de configuración espacial de los nodos

Nodo	X	Y	Z	TX	RX	CR	NCR
1	15	0	0	2	0	0	0
2	45	0	0	0	1	0	0
3	45	15	0	4	0	0	0
4	0	0	0	0	3	0	0
5	30	10	0	0	0	1	0
6	15	20	0	0	0	0	0
7	15	0	0	0	0	0	1

Para la configuración del canal también se hace uso de un fichero de texto en forma de matriz, en el que 0 representa un canal ideal, un 1 que dicho enlace no existe y un 5 que se le aplicaría la FER que aparece en el fichero de configuración. La Tabla 4.2 muestra un ejemplo ilustrativo, correspondiente al escenario de la Figura 4.3, en el que los enlaces por los que los nodos d_1 y d_2 'escuchan' los paquetes nativos de s_2 y s_1, respectivamente, tienen una FER según se establezca en el fichero de configuración.

Tabla 4.2: Fichero de configuración del canal

Nodo	1	2	3	4	5	6	7
1	1	0	0	5	0	0	0
2	0	1	5	0	0	0	0
3	0	5	1	0	0	0	0
4	5	0	0	1	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0
7	0	0	0	0	0	0	1

Para realizar la configuración del encaminamiento entre nodos, en caso de que se utilice una estrategia estática, se sigue un procedimiento parecido a los anteriores. Hay que definir, para cada nodo, el destino, próximo salto e interfaz, indicando todas las posibles rutas. Con escenarios más complejos, esta asignación de rutas y, por tanto, se optó por una estructura más sencilla en la que se indica el camino seguido por cada una de las rutas activas. En la Tabla 4.3 se muestra esta alternativa para las rutas de la topología de la Figura 4.3.

Tabla 4.3: Fichero de configuración para el encaminamiento estático

1	6	5	2
3	5	7	4

Por último, el fichero de configuración de los parámetros de simulación consta de varias etiquetas. En el se establecen, entre otras cosas, si se trabaja bajo el protocolo TCP o NC, la cantidad de datos enviados, el *Coding Time*, *Buffer Size*, *Overhearing Buffer* y el tipo de trazas que se necesita obtener para estudiar los resultados.

5

Simulaciones y resultados

En este capítulo se presentan inicialmente los parámetros que se evaluarán según los diferentes resultados de las simulaciones, describiendo el procedimiento seguido para obtenerlos.

5.1. Pasos previos

La primera tarea que se llevó a cabo, antes de implementar el algoritmo, fue acometer la generación de escenarios aleatorios. Para desplegar los nodos sobre una superficie $l \times l$ se asigna la posición x e y de cada nodo independientemente, siguiendo una distribución uniforme entre $[0, l]$. Además, para generar escenarios más proclives a utilizar NC se disponen los nodos transmisores y receptores de forma que ambos flujos de información se crucen. Con todo ello, se desarrolló un código, en C++, que siguiese los siguientes pasos:

1. Desplegar N nodos en un escenario cuadrado de $l \times l$, siguiendo un *Poisson Point Process*.
2. Obtener matriz de adyacencia.
3. Comprobar si el grafo generado es conexo.
4. Descartar escenarios no conexos.

5. Se establecen los nodos s_1, s_2, d_1, d_2, como aquellos más cercanos a las coordenadas (80,20), (80,80), (20,20) y (80,20) respectivamente.

Hay que fijar además ciertos parámetros, como el área de cobertura de cada nodo; por simplicidad se escoge el mismo para todos, utilizando un modelo de disco; También se especifica el número de nodos y el área del escenario. No se considera movilidad en los nodos, por lo que se mantendrán de forma estática durante toda la simulación, en las posiciones que se asignan durante la generación del escenario. Los parámetros que se han empleado en las simulaciones realizadas son los siguientes.

- Radio de cobertura: 20 m
- Número de nodos, N: 32
- Lado del escenario, l: 100 m

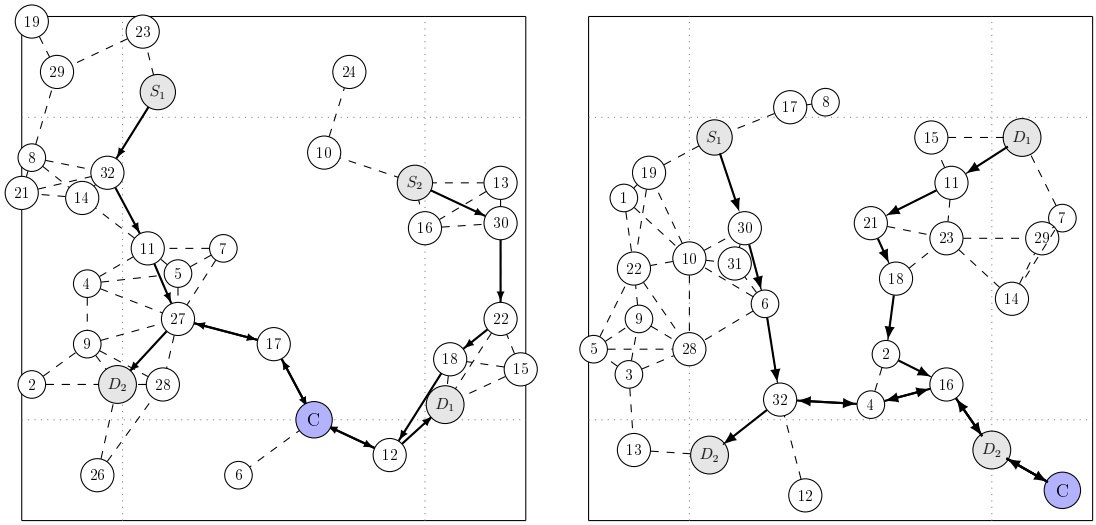
Tal como se comentó en el Capítulo 1 algunos escenarios se llevaron al simulador con objeto de comprobar el rendimiento obtenido y compararlo con el que ofrece TCP. El parámetro utilizado para medir el rendimiento es el Throughput, que se define como el cociente entre los datos útiles correctamente recibidos en el equipo destino durante la transmisión por la duración de la misma.

$$Throughput = \frac{\text{Total bits de datos}}{\text{Tiempo de simulación}} \tag{5.1}$$

5.2. Resultados de los algoritmos

Para evaluar los algoritmos se han generado 1000 escenarios con las condiciones mencionadas en la sección anterior. El Algoritmo 1 recibe la matriz de adyacencia de cada escenario y retorna la lista de los potenciales *Coding Nodes*. Se modifica además el algoritmo para que retorne la longitud de los K caminos de ambos flujos, en caso de que en el escenario exista un potencial *Coding Node*. Estos resultados se han comparado con las rutas que se obtendrían bajo un esquema tradicional, esto es, tras aplicar Dijsktra a los dos pares de nodos de comunicación respectivamente.

Para ilustrar lo que se pretende evaluar, en la Figura 5.1 se muestran dos de los escenarios generados aleatoriamente. La figura de la izquierda corresponde a una situación en la cual el *Coding Node* identificado por el algoritmo parece ser apropiado, puesto que el nodo está situado en un punto de cruce entre los dos flujos de comunicación, sin incrementar el número de saltos frente a las rutas más cortas que existen entre los dos pares de nodos



(a) Escenario con buenas características

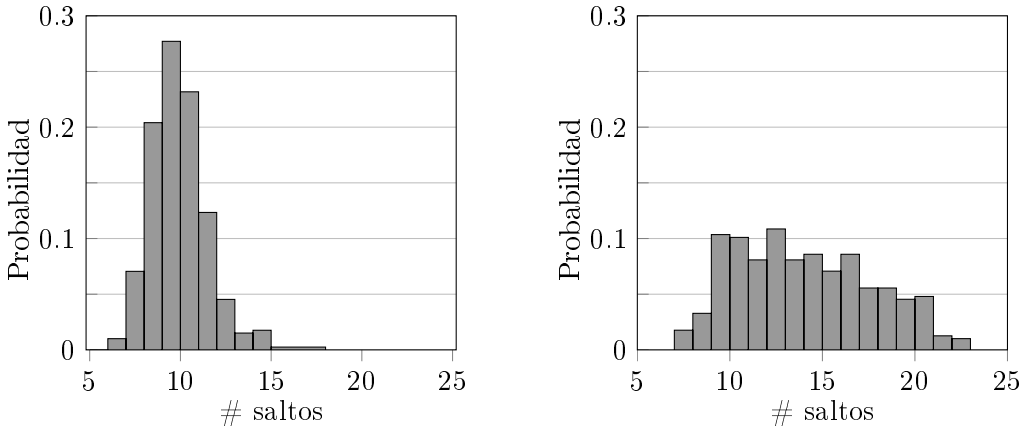
(b) Escenario descartable

Figura 5.1: Ejemplo de los resultados del Algoritmo 1

($s_1 \rightarrow d_1$, $s_2 \rightarrow d_2$). Además, la naturaleza *Broadcast* del medio inalámbrico permite que los nodos d_1 y d_2 'escuchen' directamente los paquetes nativos del flujo contrario, en concreto de los nodos 18 y 27, respectivamente. Por otro lado, la figura de la derecha presenta un comportamiento que se podría calificar como no deseable. Las rutas establecidas entre los nodos origen y destino tienen un número de saltos innecesario (por ejemplo, 2 saltos extra para el camino $s_1 \rightarrow d_1$, y 5 para el otro), lo que supone un aumento de las retransmisiones y, por tanto, una pérdida del rendimiento global de la red. Tras estos primeros resultados se puede ver que, aunque el algoritmo es capaz de encontrar el *Coding Node* más apropiado en cada escenario, no siempre proporciona una alternativa interesante desde el punto de vista del rendimiento de la red. Se requiere un segundo análisis y descartar aquellos escenarios en los que el número de saltos exceda en cierta cantidad a las rutas originales.

El primer aspecto estudiado es la función densidad de probabilidad del número de saltos requeridos por el nodo origen (s_i) para alcanzar su correspondiente destino (d_i), tal como se muestra en la Figura 5.2. Se puede ver que el esquema tradicional es siempre capaz de encontrar rutas con un número de saltos ≤ 15, situandose los valores más probables en le intervalo $9-11$ saltos; por su parte, con el esquema de NC se se observa una distribución más uniforme. Esto quiere decir que, en algunos escenarios, el esquema de rutas empleado para aplicar NC impone rutas con un número de saltos mucho mayor, tal y como se ha comentado en el párrafo anterior.

Los resultados que se presentan se obtienen utilizando el Algoritmo 1 y, por tanto, existe una cierta dependencia con el parámetro K. En la Figura 5.3 puede observarse el mismo resultado para diferentes valores de K. No existe prácticamente ninguna diferencia, salvo que, a medida que se aumenta el parámetro K, aparecen escenarios que pueden aplicar NC estableciendo rutas con un número de saltos. Esto se observa mejor en la Figura 5.3d



(a) Esquema tradicional (K=1)

(b) Network coding (K=1)

Figura 5.2: Función densidad de probabilidad del número de saltos

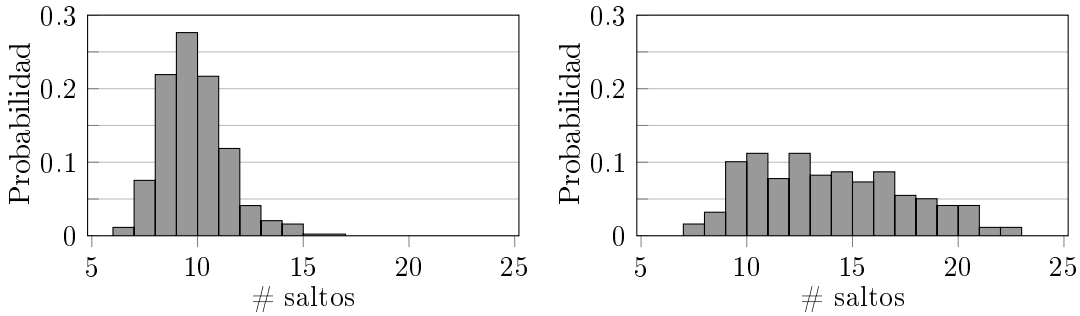
donde se generan escenarios con rutas cercanas a 25 saltos. En estas circunstancias, al examinar rutas alternativas que, en general, tienen más saltos, se facilita la aparición de escenarios como los que se muestran en la Figura 5.1b

Otra forma de observar que, por norma general, el uso de NC implica la utilización de rutas con mayor número de saltos se presenta en la Figura 5.4, que compara el número de saltos de la ruta con mayor longitud obtenida con NC frente al esquema original TCP. En el eje de abscisas se representa el número de saltos de la ruta con mayor número de saltos (para los dos flujos) y en el eje de ordenadas el número de saltos utilizando NC. No se presentan en este caso los resultados para distintos valores de K ya que, al igual que en el caso anterior, no varían.

Por último se estudia la densidad de probabilidad de que un escenario aleatorio pueda utilizar NC en función del parámetro Δ, que restringe la probabilidad establecer una configuración válida de NC, limitando el número de saltos que pueden exceder las rutas establecidas con NC frente al esquema original. En la Figura 5.5 se muestra el resultado en función del parámetro K del algoritmo; se puede ver que, incrementando el valor de K, se aumenta la probabilidad de poder aplicar NC en el escenario. Los resultados muestran que se puede establecer NC en $\approx 47\,\%$ de los casos (cuando no se limita Δ).

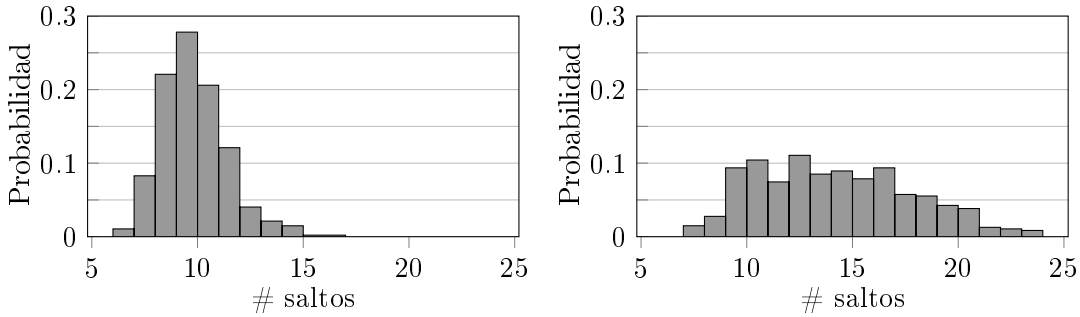
Como se expone se verá posteriormente, un valor de Δ superior a 1 deriva en un empeoramiento del el rendimiento que se obtendría utilizando el protocolo TCP original. Bajo la restricción $\Delta = 1$ la probabilidad de poder aplicar NC sobre un escenario aleatorio es inferior al $10\,\%$ ($\approx 5\,\%$ en el caso de $\Delta = 0$).

Con intención de comprobar el porcentaje de escenarios válidos en el caso de $\Delta = 0$ (que es, además, el caso más favorable para el uso del esquema NC), se usa el Algoritmo 2, obteniendo que solo en un $5.5\,\%$ de los escenarios es posible usar NC.



(a) Esquema tradicional

(b) Network coding (K=2)



(c) Esquema tradicional

(d) Network coding (K=3)

Figura 5.3: Función densidad de probabilidad del número de saltos

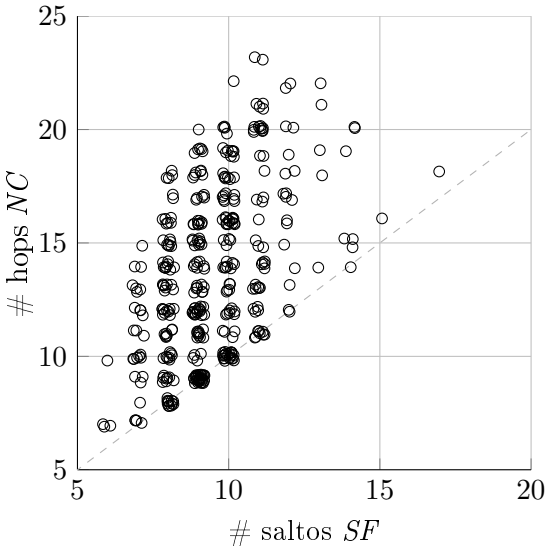


Figura 5.4: Comparativa entre las rutas de mayor longitud del esquema tradicional frente NC

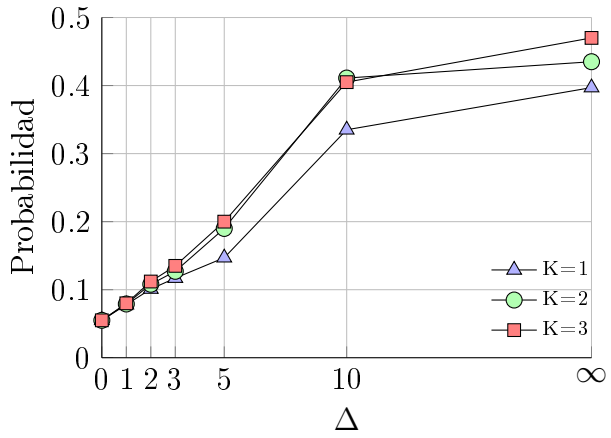


Figura 5.5: Probabilidad de encontrar al menos un *Coding Node*

5.3. Resultados de Simulación

Con intención de analizar el rendimiento que ofrece el esquema NC sobre redes WMN se han exportado los escenarios generados al simulador ns-3. Primero se deben escoger los parámetros de configuración de NC expuestos en la Sección 2.2. Para su elección se han simulado los escenarios bajo diferentes configuraciones, viendo cuál era la que ofrecía mejor rendimiento. A continuación se exponen los valores de los parámetros que se han empleado durante la simulación.

- Número de paquetes: 5000 paquetes
- Longitud de los paquetes: 1432 bytes
- IEEE 802.11b en todos los enlaces: 11 Mbps
- Todos los enlaces son ideales.
- *Buffer Size*: 100 paquetes
- *Coding Time*: 100 paquetes
- Máximo número de paquetes codificados: 2 paquetes

Inicialmente se analizan los escenarios generados con $\Delta = 0$, aquellos que, sin alterar las rutas originales, pueden aplicar NC. En la Figura 5.6 se presenta la función de densidad de probabilidad acumulada del throughput comparando que se obtiene bajo NC y utilizando TCP. En el 50 % de los casos se obtiene un throughput de ≈ 0.66 Mbps bajo NC, mientras que bajo el esquema original se obtiene un ≈ 0.65, lo que supone una mejora de un 1.5 %.

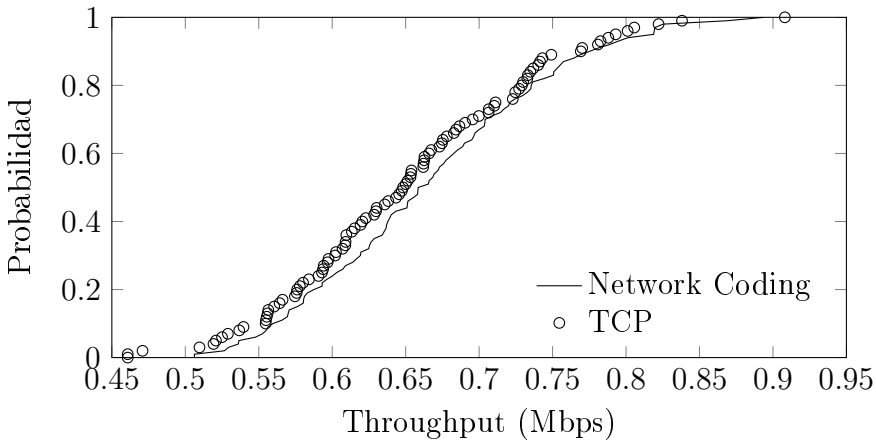


Figura 5.6: Densidad de probabilidad del Throughput bajo $\Delta = 0$

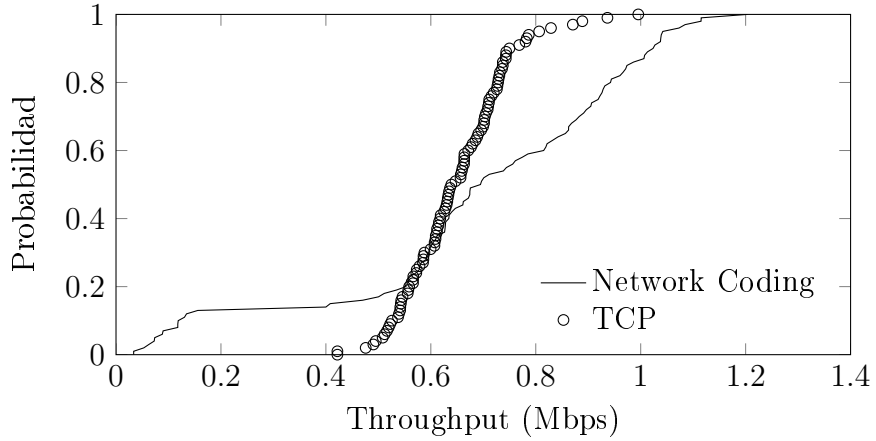


Figura 5.7: Densidad de probabilidad del Throughput bajo $\Delta = 1$

Comparando el valor medio del rendimiento se pasa del ≈ 0.67 con NC a ≈ 0.65 con TCP, con una mejora del 2 % aproximadamente.

En un segundo grupo de simulación, se utilizan escenarios generados tras relajar la restricción de Δ, $\Delta = 1$. Los resultados, que se muestran en las Figura 5.7,ponen de manifiesto que no siempre es rentable el uso del esquema NC. Existe, en un amplio porcentaje de los escenarios, una pérdida de rendimiento frente al esquema original. En estos casos la mejora que puede suponer el uso de NC no compensa la pérdida de rendimiento que acarrea tener un salto más en cada flujo. Sin embargo, el throughput medio mejora de ≈ 0.65 bajo TCP a ≈ 0.69 bajo NC, una mejora de casi 6 %.

Para analizar los motivos por los que, en el caso de $\Delta = 1$, la mejora del rendimiento es superior que con $\Delta = 0$ se debe estudiar el número se saltos en ambas configuraciones. Al incrementar el valor de Δ se observa que los escenarios obtenidos tienden a tener menor número de saltos. En la Figura 5.8 se muestra la distribución del número medio de saltos

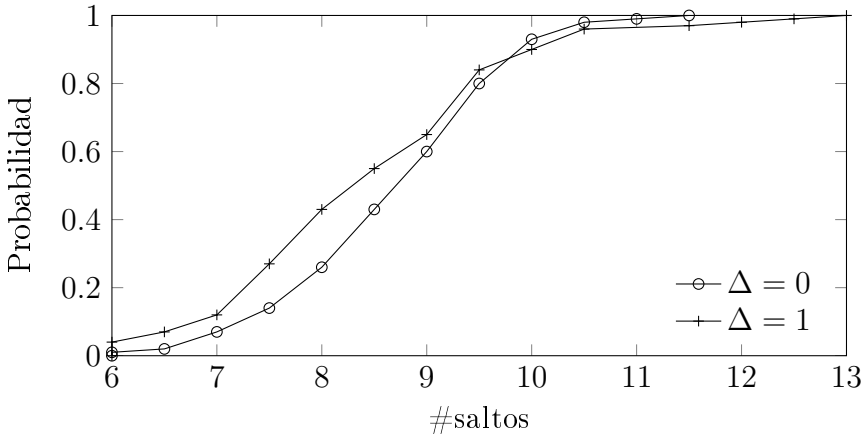


Figura 5.8: Densidad de probabilidad del número de saltos medio

para escenarios generados con $\Delta = 0$ y $\Delta = 1$. Al existir, en algunos casos, un menor número de saltos, la mejora que introduce NC es superior; esto es, ahorrar un salto codificado en una ruta de 6 saltos supone una mayor mejora relativa que un salto codificado en un camino de 11 saltos.

Bajo escenarios generados con relajación mayor del parámetro Δ el esquema NC no es capaz de ofrecer ninguna mejora en el rendimiento.

6

Conclusiones y líneas futuras

En este último capitulo se recogen las principales conclusiones que se han ido recopilando durante la realización del trabajo, haciendo especial hincapié en aquellas obtenidas tras analizar los resultados presentados en el Capítulo 5. Finalmente se expondrán posibles líneas futuras de investigación que han quedado abiertas a raíz del estudio llevado a cabo.

6.1. Conclusiones

Como consecuencia del cambio tan sustancial que se ha producido en el mundo de las comunicaciones e Internet la comunidad investigadora ha desarrollado propuestas con las que abordar la demanda del creciente número de usuarios que, a su vez, cada vez necesita mayor capacidad, especialmente en las comunicaciones inalámbricas. Las redes malladas suponen un ahorro importante al cubrir extensas áreas de cobertura, facilitando además la incorporación de nodos a la red. Sin embargo, el rendimiento que se obtiene sobre ellas es escaso. Como solución se ha propuesto utilizar técnicas de codificación de flujos, Network Coding, que pretende mejorar el rendimiento que se puede alcanzar sobre redes malladas inalámbricas.

Muchos estudios han cubierto varios aspectos trabajando, por norma general, sobre topologías sencillas, con intención de comprender los diferentes parámetros de operación que afectan al rendimiento, como por ejemplo: *Buffer size*, *Coding Time*, etc. Sin embargo, este trabajo pretende estudiar la posibilidad de aplicar dichas técnicas.

Se parte de las condiciones de codificación propuestas en [[12], [13]] que, se adaptan para trabajar con el protocolo de NC disponible, que impone que el nodo decodificador debe ser además destino de alguno de los flujos. Finalmente se desarrollan dos algoritmos que se centran en identificar la existencia de *Coding Nodes*.

El primer algoritmo permite conocer los potenciales *Coding Nodes* y las rutas que se deben configurar sobre el escenario para aplicar NC. Tiene como limitación una dependencia con el número de rutas que se calculan. Para solventar ese problema se desarrolla otro algoritmo que, aunque no devuelve las rutas a configurar, permite conocer el número exacto de escenarios donde se puede aplicar NC bajo las condiciones establecidas.

Los resultados muestran una baja probabilidad de utilizar NC sobre topologías aleatorias, a pesar de haber desplegado los escenarios con intención de que los flujos de comunicación se crucen. Únicamente el 47 % de los escenarios estudiados muestran posibilidad de aplicar NC. Además, en la mayoría de los casos, el esquema de rutas obtenido descartaría en el uso de NC, en tanto en cuanto el número de saltos de dichas rutas es muy superior al que se obtendría utilizando un esquema más tradicional. Si se limita el parámetro $\Delta = 0$, que marca un máximo al número de saltos adicionales para las rutas obtenidas con NC frente a la situación original, el número de escenarios con posibilidad de aplicar codificación baja al 5 %. La aplicabilidad de estas técnicas son escasas.

Además del reducido número de escenarios sobre los que se puede utilizar el esquema de NC, el rendimiento que se obtiene no muestra un gran incremento. En el mejor caso se obtiene una mejora del 5 %, bajo $\Delta = 1$. Cabe tener en cuenta que establecer el parámetro $\Delta = 1$ no siempre se obtiene una mejora en el rendimiento, como se observa en la Sección 5.3, mientras que eso sí se puede asegurar con $\Delta = 0$, al menos bajo condiciones ideales. Pero en este caso la mejora del rendimiento es de un 2 % (valor medio).

Como parte adicional al trabajo se ha estudiado qué mejora se obtendría si se eliminase la restricción del nodo decodificador. Los resultados son positivos, y permitirían utilizar NC en un 40 % de los casos, limitando el valor de $\Delta = 0$. En estas configuraciones no se puede comprobar el rendimiento que se obtendría bajo esta situación, aunque cabe esperar que no presentaría una variación muy apreciable.

El primer algoritmo y los resultados presentados en la sección 5.2 se han aprovechado para enviar un artículo a la conferencia internacional MONAMI, actualmente en proceso de revisión. El título de dicha publicación es 'On the Feasibility of Inter-Flow Network Coding over Random Wireless Mesh Network'

6.2. Líneas futuras

Aunque son numerosos los estudios que han tratado el tema de las técnicas de Network Coding, existen todavía muchos aspectos abiertos a estudios futuros. A continuación se exponen algunos temas que se podrían tratar en trabajos posteriores.

En primer lugar sería interesante generalizar el algoritmo para considerar un mayor número de flujos. La idea de combinar las redes malladas con NC implica la existencia de multitud de flujos de información, con numerosos puntos de cruce, esto es, potenciales *Coding Nodes*. El algoritmo debería ser capaz de detectar esos *Coding Nodes* pero, además, se debería estudiar qué combinaciones de flujos son las que darían lugar a una mejora en el rendimiento.

También se podría intentar evaluar el funcionamiento de NC bajo un comportamiento dinámico. Es decir, la disposición de los nodos varía y los flujos de información podrían iniciarse o finalizar entre cualquier par de nodos de manera aleatoria. Se podría estudiar así la aplicabilidad de estas técnicas sobre situaciones más realistas.

En cuanto a la simulación de los escenarios, se debería estudiar bajo condiciones no ideales del canal. Además, podrían añadirse flujos de información independientes a lo largo del escenario, con objeto de estudiar la repercusión de los fallos de codificación debidos a la pérdida de paquetes nativos e interferencias de otros flujos.

Bibliografía

[1] Jangeun Jun and M.L. Sichitiu. The nominal capacity of wireless mesh networks. *Wireless Communications, IEEE*, 10(5):8–14, Oct 2003.

[2] M. Zorzi, A. Chockalingam, and R.R. Rao. Throughput analysis of tcp on channels with memory. *Selected Areas in Communications, IEEE Journal on*, 18(7):1289–1300, July 2000.

[3] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R.W. Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204 –1216, July 2000.

[4] S. Katti, H. Rahul, Wenjun Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the Air: Practical Wireless Network Coding. *Networking, IEEE/ACM Transactions on*, 16(3):497 –510, 2008.

[5] R. Koetter and M. Medard. An algebraic approach to network coding. In *Information Theory, 2001. Proceedings. 2001 IEEE International Symposium on*, page 104, 2001.

[6] Zheng Liu and Shudong Jin. Diagnosing the limitations of network coding at transport layer. In *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, pages 436–442, 2010.

[7] Yunnan Wu. Information exchange in wireless networks with network coding and physical-layer broadcast. 2004.

[8] M. Hunderbll, J. Ledet-Pedersen, J. Heide, M.V. Pedersen, S.A. Rein, and F.H.P. Fitzek. CATWOMAN: Implementation and Performance Evaluation of IEEE 802.11 Based Multi-Hop Networks Using Network Coding.

[9] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich. ABetter Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.). IETF Internet Draft – work in progress 07, individual, April 2008.

[10] David Gómez, Sofiane Hassayoun, Arnaldo Herrero, Ramón Agüero, and David Ros. Impact of network coding on TCP performance in wireless mesh networks. In *23th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE Proceedings*, September 2012.

[11] David Gómez, Sofiane Hassayoun, Arnaldo Herrero, Ramón Agüero, David Ros, and Marta García-Arranz. On the addition of a network coding layer within an open connectivity services framework. In *4th International Conference on Mobile Networks and Management (MONAMI), 2012*, September 2012.

[12] Jilin Le, J.C.-S. Lui, and Dah-Ming Chiu. DCAR: Distributed Coding-Aware Routing in Wireless Networks. *Mobile Computing, IEEE Transactions on*, 9(4):596–608, April 2010.

[13] Bin Guo, Hongkun Li, Chi Zhou, and Yu Cheng. Analysis of general network coding conditions and design of a free-ride-oriented routing metric. *Vehicular Technology, IEEE Transactions on*, 60(4):1714–1727, May 2011.

[14] E. W. Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.

[15] Jin Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

Lista de acrónimos

WMN Wireless Mesh Network

NC Network Coding

BATMAN Better Approach To Mobile Ad-hoc Networking

CATWOMAN Coding Applied To Wireless On Mobile Ad-hoc Networks

CRM Coding-aware Routing Metric

DCAR Distributed Coding-Aware Routing

BFS Breadth-First Search

DFS Depth-Firts Search

FORM Free-ride Oriented Routing Metric

LBNL Lawrance Berkely National Laboratory

Tcl Tool Command Language

OTcl Object Tool Command Languaje